Unit-4

Push down Automata (PDA)

→ Push down automata is implementation of CFG

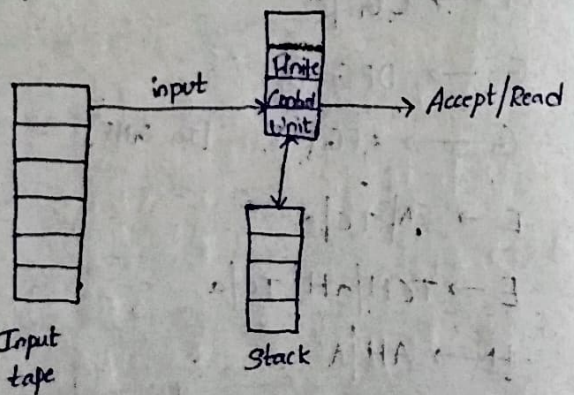
→ PDA accepts the context free grammer

→ The language generated by context free grammer is context free language.

→ PDA has 7-tuple representation.

→ PDA is faster than finite automata.

Model of PDA



→ Input tape is divided into many sub ka cell.

→ In each cell one symbol is to be placed.

→ Pushdown automata vreads one symbol at a time.

→ The finite control unit has a pointer, which points the current symbol which is to be vread

→ Stack is a structure used to store the items temporarily.

→ By using stack, we can push and remove the items from one end only

→ Stack has an infinite size.

→ Stack uses last in first out while performing the push and pop operations.

→ PDA = Finite automata + stack.

→ Finite automata can vember only finite amount of information but Pushdown automata can vemember infinite amount of information.

→ A pushdown automata is more powerful than finite automata.

→ Any language which can be acceptable by finite automata is also acceptable by pushdown automata.

→ Pushdown automata also accepts a class of language which even cannot be accepted by finite automata. Therefore, pushdown automata is much more superior than finite automata.

→ eg:- 0^n1^n → finite automata cannot be designed by the language 0^n1^n. But pushdown automata which can accept the language 0^n1^n.

→ Finite state machine has a very limited memory. But pushdown automata has more memory, with the help of a stack component.

Representation of Pushdown Automata

→ Pushdown automata is formally defined by 7 tuples as shown below.

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

Where,

Q = A finite set of states

Σ = A finite set of input symbols.

Γ = A finite stack alphabet.

δ = A transition function

q_0 = Initial state

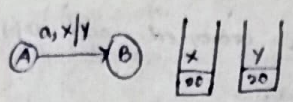
Z_0 = Initial stack symbol

F = A finite set of final states.

D/b finite automata and Pushdown automata.

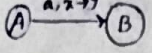
Finite automata	PDA
→ Finite automata can remember only limited amount of information	→ PDA can remember large amount of information
→ Not contain storage element like stack	→ Contain storage element i.e stack.
→ 5 tuple representation	→ 7 tuple representation.
→ Result 5 tuple	→ Two types of PDA
Two types of FA	- deterministic PDA
- DFA	- Nondeterministic PDA
- NFA	

Graphical representation

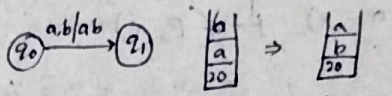


Here 'a' is input and then x is poped and Y is pused.

(or)



Ex:



Here $\delta(q_0, a, b) = (q_1, ab)$

b is poped,
b, a is pushed.

Model of PDA

Instantanious description

→ Instantanious description is an informal decision who a PDA computs an input string and make a decision that string is accepted or rejected.

→ An instantanious description is a triple notation.

(q, w, a)

q is current state

w is remaining input string

a represents stack contents.

$\vdash$ (turnstile notation) it describes one move.

$\vdash^*$ represents multiple move.

eg: $(P, \underset{\text{current state}}{b}, \underset{abba}{T}) \vdash (q, \underset{\text{next state}}{w}, \underset{bba}{a})$

While taking transition from P to q the input symbol from b is taken and the top of stack is removed and replaced by a new string α.

eg: 1) $(q, aw, x\beta) \vdash (P, w, \alpha\beta)$

2) $(q, a, x) \vdash (P, \alpha)$ (right, stack)

Accepting of the input string by the PDA (final state).

Let $A = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$

$T(A) = \{w \in \Sigma^* / (q_0, w, z_0) \vdash^* (q_F, \epsilon, \alpha)\}$

$q_F \in F$ (belongs) and $\alpha \in \Gamma^*$ $(q_0, w, \alpha) \vdash^* (q_F, \epsilon, \alpha)$

Acceptance by empty stack

Let $A = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$ be a the pushdown automata the set

$T(A)$ accepted by empty stack is designed by

$T(A) = \{w \in \Sigma^* / (q_0, w, z_0) \vdash^* (q_p, \epsilon, \epsilon)\}$

$q_p \in Q$ $(q_0, w, \alpha) \vdash^* (q, \epsilon, \epsilon)$

Types of PDA

1. deterministic PDA
2. Non deterministic pushdown automata

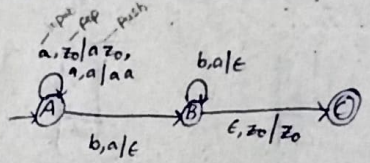
Deterministic Pushdown Automata

The PDA that has almost one choice of move in any state that is called deterministic Pushdown automata.

Let $A = \{Q, \Sigma, \delta, q_0, F, \Gamma, z_0\}$

transition function $\delta = Q \times \{\Sigma \cup \epsilon\} \times \Gamma \to Q \times \Gamma^*$

ex:



Let input string be aabb.

$A(a$

$(A, aabb, z_0)$ Initially

$\vdash (A, abb, az_0)$

$\vdash (A, bb, aaz_0)$

$\vdash (B, b, az_0)$

$\vdash (B, \epsilon, z_0)$

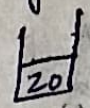
$\vdash (B, \epsilon, z_0)$ z_0 is final stack content.

Design PDA for acceptance by final state

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

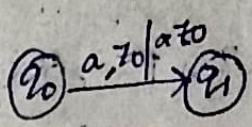
$\{ab, aabb, aaabbb, \ldots\}$ [No. of a's followed by No. of b's]

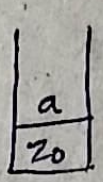
Initially stack contents is Z_0



Now

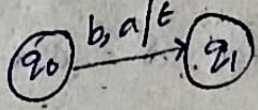
push a

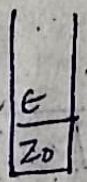




push a

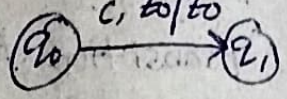
pop a



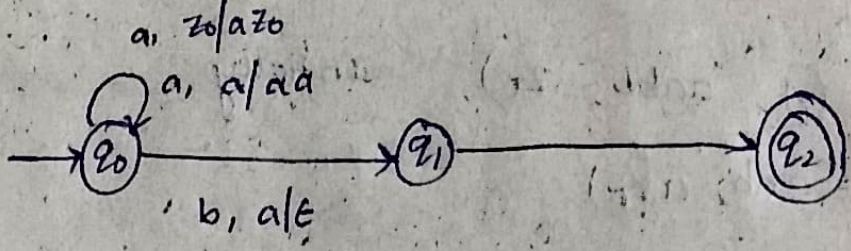


② push ϵ,
① pop a

No change



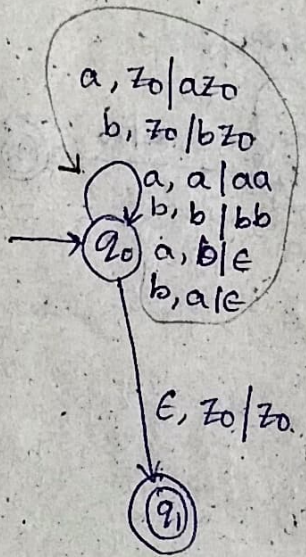
② push Z0,
① pop Z0



d. $S \rightarrow aAbBbC / aB / bC$.

Design PDA for the set of all strings having equal no. of a's and no. of b's.

$\{ab, abab, abba, ba, \ldots\}$

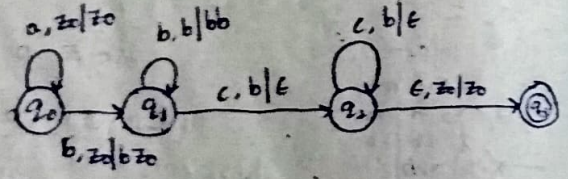


(b)
b
z0

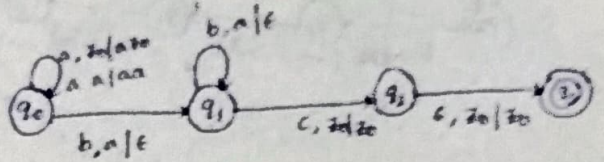
.. string can start with a (or) can start with b.

Design PDA for $L = a^n b^n c^n \mid n \geq 1$

$\{aaabc, aaabbcc, aaabbbccc, \cdots\}$

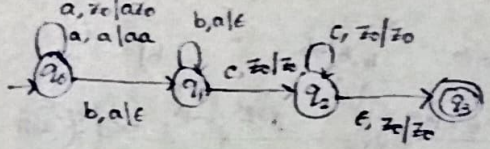


$L = a^n b^n c \mid n \geq 1$



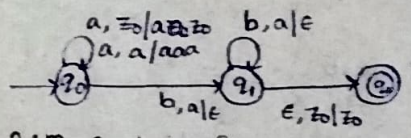
$L = \{a^n b^n c^m \mid m, n \geq 1\}$

$\{abc, aabbc, \cdots\}$



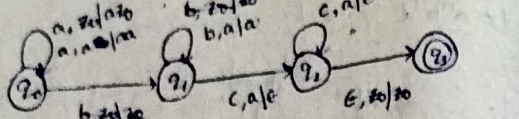
$L = \{a^n b^{2n} \mid n \geq 1\}$

$\{abb, aabbbb, \cdots\}$

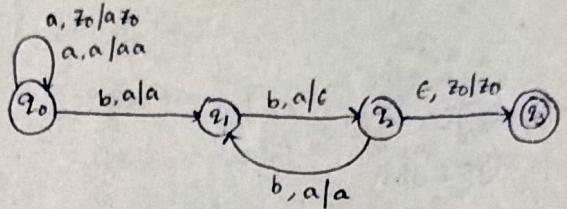


$L = \{a^n b^m c^n \mid n \geq 1\}$

$\{abc, aabcc, \cdots\}$

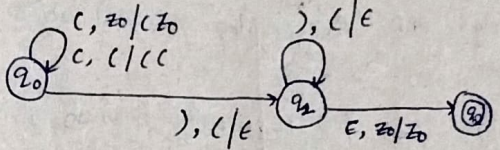


$a^n b^m c^n$

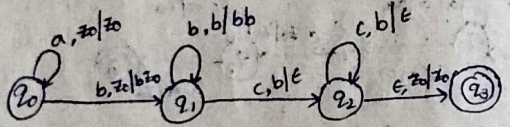


Design PdA for balancing of parenthsis

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



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

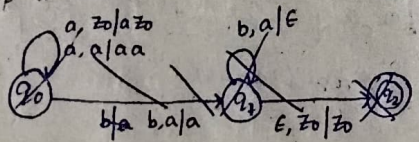


Design $L = \{a^n b^{2n} \mid n \geq 1\}$

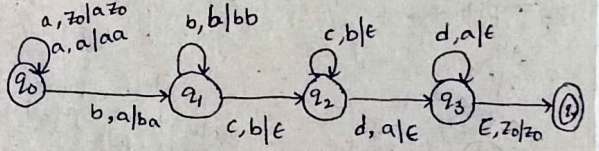
a → push a

b → No change for 1b

b → pop a



Design PdA for $L = \{a^n b^m c^m d^n \mid m, n \geq 1\}$.



Design PDA for $L = \{wc\,w^R\}$ over $\Sigma = \{a, b, c\}$ where $w = (a+b)^*$ (or) odd length palindrome.

R → reverse of the string.

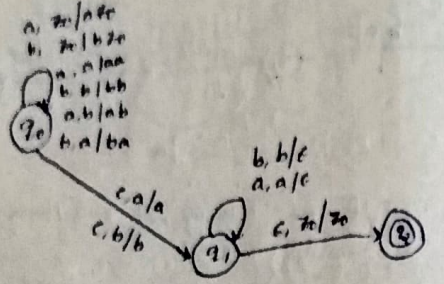
If top = input symbol after c then pop the top.

eg: $w = (a+b)^*$

eg: $w = abab$

$w^R = baba$ (top / pop)

then $L = ababcbaba$

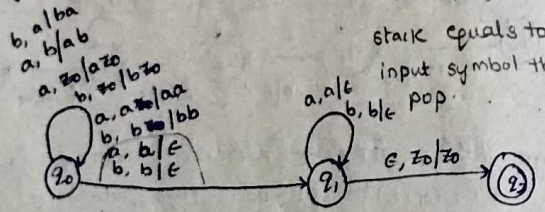


Design PDA for (OR) accepting the odd length palindrome over $\Sigma = \{a, b, c\}$.

Design PDA for $L = \{w w^R\}$ over an $\Sigma = \{a, b\}$ where $w = (a+b)^*$

(OR)

Design PDA which accept the strings of even length palindromes.



If top of w in stack equals to input symbol then pop.

***** It is non deterministic PDA. depending on input, top and stack contents transition takes place.

String acceptance:-

$\delta(q_0, abba, Z_0)$

$\vdash (q_0, bba, aZ_0)$

$\vdash (q_0, ba, baZ_0)$

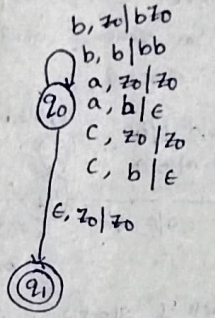
$\vdash (q_0, a, bbaZ_0)$ [it is NPDA so goes to q_1]

$\vdash (q_1, a, aZ_0)$

$\vdash (q_1, \epsilon, Z_0)$

$\vdash (q_2, \epsilon, \epsilon)$.

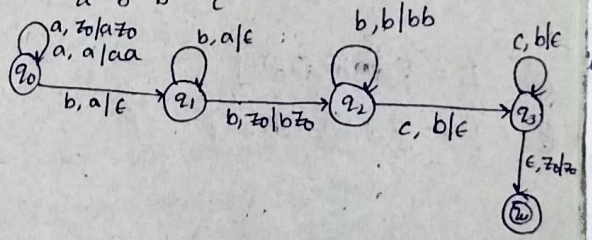
Design $L = \{a^n b^{n+m} c^m \mid m, n \geq 1\}$.



(doubt)

(OR)

$a^n b^n b^m c^m$



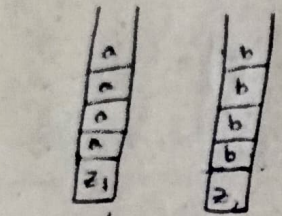
Two stack PDA

Short style representation

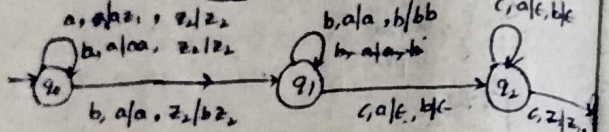
$(Q, \Sigma, \Gamma_1, \Gamma_2, \delta, q_0, z_1, z_2, F)$

$\delta(q_0, a, z_1, z_2) = (q_0, az_1, z_2)$

Q Design PDA with two stack $L = \{a^n b^n c^n\}$.



whenever C comes pop 'a' in stack 1 and 'b' in stack 2.



When we add 'a' in stack 1 stack 2 is work

" " " b " " 2 " 1 " "

(a) $L = \{a^m b^n a^n b^n\}$

(b) $L = \{a^n b^n c^n d^n e^n\}$

Conversion from CFG to PDA

Design PDA for given CFG

Procedure :-

1. Convert the given production of CFG into GNF.
2. The PDA will only have one state 'q'.
3. The initial symbol of CFG will be 'S'.
4. that is the initial symbol in the PDA.
5. For each non terminal add the following rule. i.e if the production

$A \longrightarrow B$

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

5. For each terminal symbol add the following rule.

$A \longrightarrow aB \quad \delta(q, a, a) = (q, \epsilon)$

$B \longrightarrow b \quad \delta(q, b, b) = (q, \epsilon)$

Design PDA for given CFG

$S \longrightarrow aAA$

$A \longrightarrow aS \mid bS \mid a$

1. four tuple representation.
2. It is already in GNF.

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

$\delta(q, \epsilon, A) = \{(q, aS)(q, bS)(q, a)\}$

$A \to BC$

$B \to b \mid \epsilon$

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

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