UNIT-4 PUSH DOWN AUTOMATA

INTRODUCITON:

- A PDA is an enhancement of finite automata (FA)
- A PDA is a way to implement a CFG in a similar way we can design FA for regular grammar.
- Finite automata with a stack memory can be viewed as Pushdown automata.
- PDA=FSM + Stack.
- Addition of stack memory enhances the capability of Pushdown automata compared to finite automata.
- The stack memory is potentially infinite and linearly arranged.

MODEL OF PUSHDOWN AUTOMATA

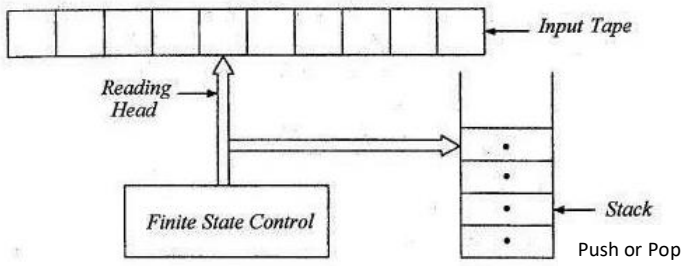


FIGURE : Model of Pushdown Automata

It consists of:

(1) a Read only finite tape

(2) a Reading head (which reads from the tape

(3) a Stack memory (operating in LIFO fashion)

There are two alphabets: one for input tape and another for stack. The stack alphabet is denoted by Γ and input alphabet is denoted by Σ . PDA reads from both the alphabets; one symbol from the input and one symbol from the stack.

MATHEMATICAL REPRESENTATION OF 'PDA':

A pushdown automata is described by 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where

1. Q is finite and nonempty set of states,
2. Σ is input alphabet,
3. Γ is finite and nonempty set of pushdown symbols,
4. $q_0 \in Q$, is the starting state,
5. $Z_0 \in \Gamma$, is the starting (top most or initial) stack symbol, and
6. $F \subseteq Q$, is the set of final states.
7. δ, is the transition function,

Here $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \rightarrow Q \times \Gamma^*$

❖ 'δ' takes three tuples as input like δ(q,a,Z) where ,

i) q is a state in Q

II) 'a' is either an input symbol in Σ or 'a' is also belongs ε.

iii) 'Z' is a stack symbol i.e member of Γ.

iv) The output of δ is finite set of pairs like (q,ϒ) where,

p: it is a new state.

ϒ: it is a set of stack symbols that replace 'z' at the top of the stack.

MOVES OF PDA:

The move of PDA means that what are the options to proceed further after reading inputs in some state and writing some string on the stack. Since PDA is non-deterministic device having some finite number of choices of moves in each situation.

The move will be of **two** types:

1. In the first type of move, an input symbol is read from the tape, it means the head is advanced and depending upon the topmost symbol on the stack and present state, PDA has number of choices to proceed further.

Mathematically first type of move is defined as follows.

$\delta(q,a,Z) = \{(p_1,\alpha_1),(p_2,\alpha_2),\ldots(p_n,\alpha_n)\}$, where for $1 \le i \le n, q, p_i$ are states in Q, $a \in \Sigma$, $Z \in \Gamma$, and $\alpha_i \in \Gamma^*$.

PDA reads an input symbol 'a' and one stack symbol **'Z',** in present state 'q' and for any value(s) of i, enters state 'p', replaces stack symbol **'Z'** by string **α** and head is advanced one cell on the tape. Now, the leftmost symbol of string α_i is assumed as the topmost symbol on the stack.

2. In the second type of move, the *input symbol is not read* from the tape, it means head is not advanced and topmost symbol of stack is used. The topmost of stack is modified without reading the input symbol. It is also known as ***ε-move***.

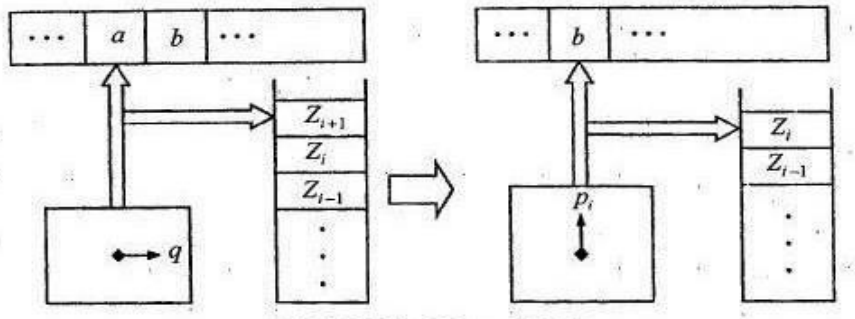
Mathematically second type of move is defined as follows.

$\delta(q,\in,Z) = \{(p_1,\alpha_1),(p_2,\alpha_2),\ldots(p_n,\alpha_n)\}$, where for $1 \le i \le n, q, p_i$ are states in Q, $a \in \Sigma$, $Z \in \Gamma$, and $\alpha_i \in \Gamma^*$.

PDA does not read input symbol but it reads stack symbol **'Z',** in present state 'q' and for any value(s) of i, enters state 'p_i' , replaces stack symbol **'Z'** by string α_i and head is not advanced on the tape. Now , the leftmost symbol of string α_i is assumed as the topmost symbol on the stack.

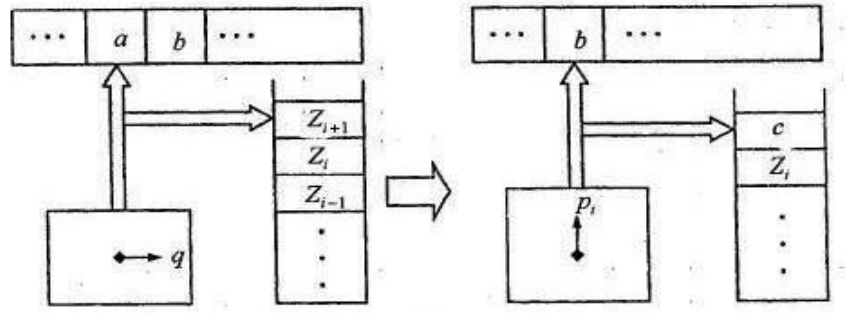
The string α_i be any one of the following:

1. $\alpha_i = \in$ in this case the topmost stack symbol Z_{i+1} is erased and second topmost symbol becomes the topmost symbol in the next move. It is shown in figure (a).



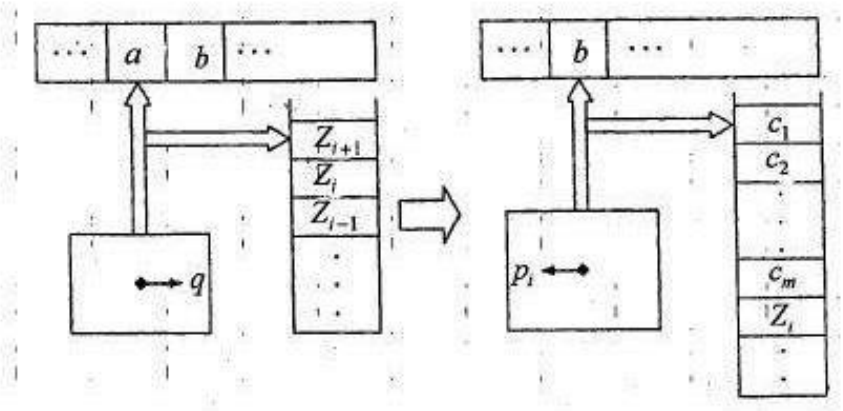
FIGURE(a): Move of PDA

2. $\alpha_i = c, c \in \Gamma$, in this case the topmost stack symbol Z_{i+1} is replaced by symbol c. It is shown in figure(b)



FIGURE(b): Move of PDA

3. $\alpha_i = c_1 c_2 \ldots c_m$, in this case the topmost stack symbol Z_{i+1} is replaced by string $c_1 c_2 \ldots c_m$. It is shown in figure(c).



FIGURE(c): Move of PDA

For example:

(1) $\delta(q,a,Z)=(p_1,\varepsilon)$

It indicates that 'Z' is erased from the stack and state is changed from q to p_1.

(2) $\delta(q,a,Z)=(p_1,cZ)$

It indicates that form state q, read input symbol 'a', where top of the stack Z. then the finite control goes to p_1 state and adding the element 'c' to the top of the stack.

(3) $\delta(q,a,Z)=(p_1,Z)$

It indicates that on reading symbol 'a' state is changing from q_1 to q_2 and there is no change in the stack (bypass operation).

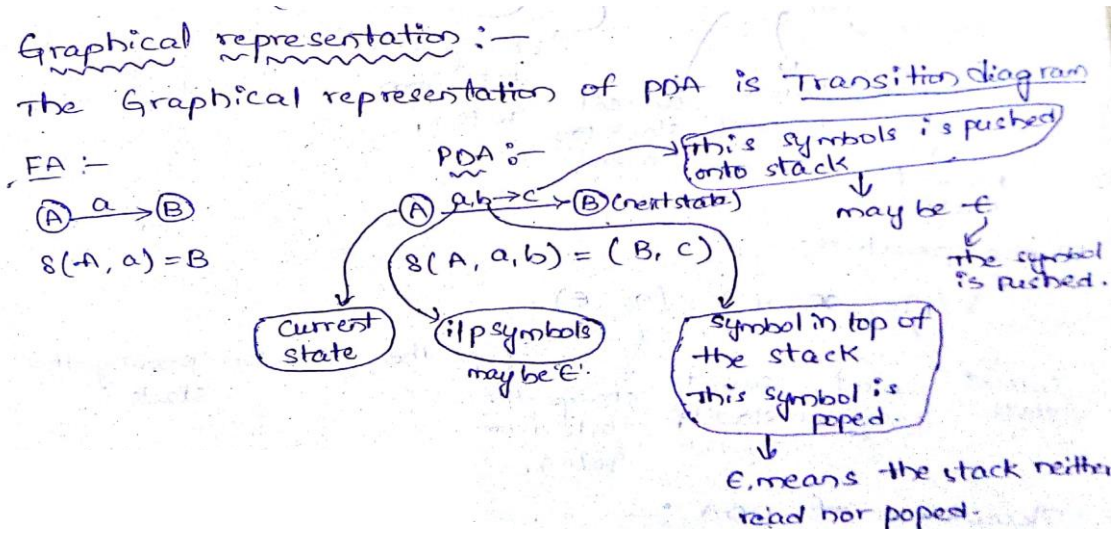
Note: let M be a PDA. A move relation denoted by ' ⊢ ' between IDs is defined as :

$(q,a_1,a_2,\ldots\ldots a_n,Z_1,Z_2,\ldots\ldots Z_m) \vdash (p_1,a_2,a_3,\ldots.a_n,cZ_2,Z_3,\ldots.Z_m)$ if $\delta(q,a_1,Z_1)$contains(q_1,c).

The above move relation can be described as follows: The PDA in state q with $Z_1,Z_2,\ldots\ldots Z_m$ in PDS (Z_1 is at the top) reads the input symbol a1. When (p1,c) ϵ $\delta(q,a_1,Z_1)$, the PDA moves to a state p1 and writes 'c' on the top of $Z_1,Z_2,\ldots\ldots Z_m$. After this transition, the input string to be processed is $a_2,a_3\ldots\ldots a_n$ and symbols on stack is $cZ_2,Z_3,\ldots.Z_m$

If $c=c_1,c_2\ldots.c_m$ then FIGURE(c) illustrates the moves.

GRAPHICAL REPRESENTATION OF PDA:

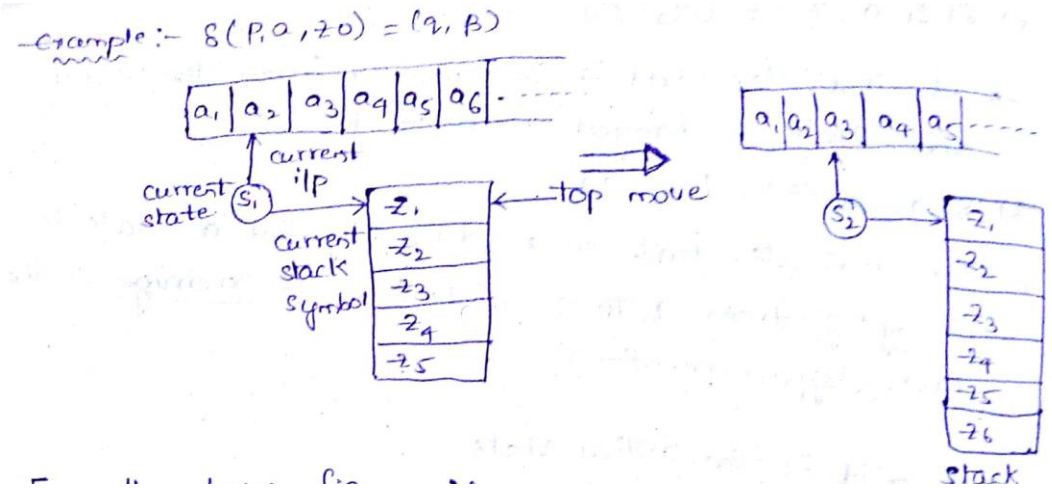


INSTANTANEOUS DESCRIPTION (ID):

It is used to describe the configuration of PDA at given instance.

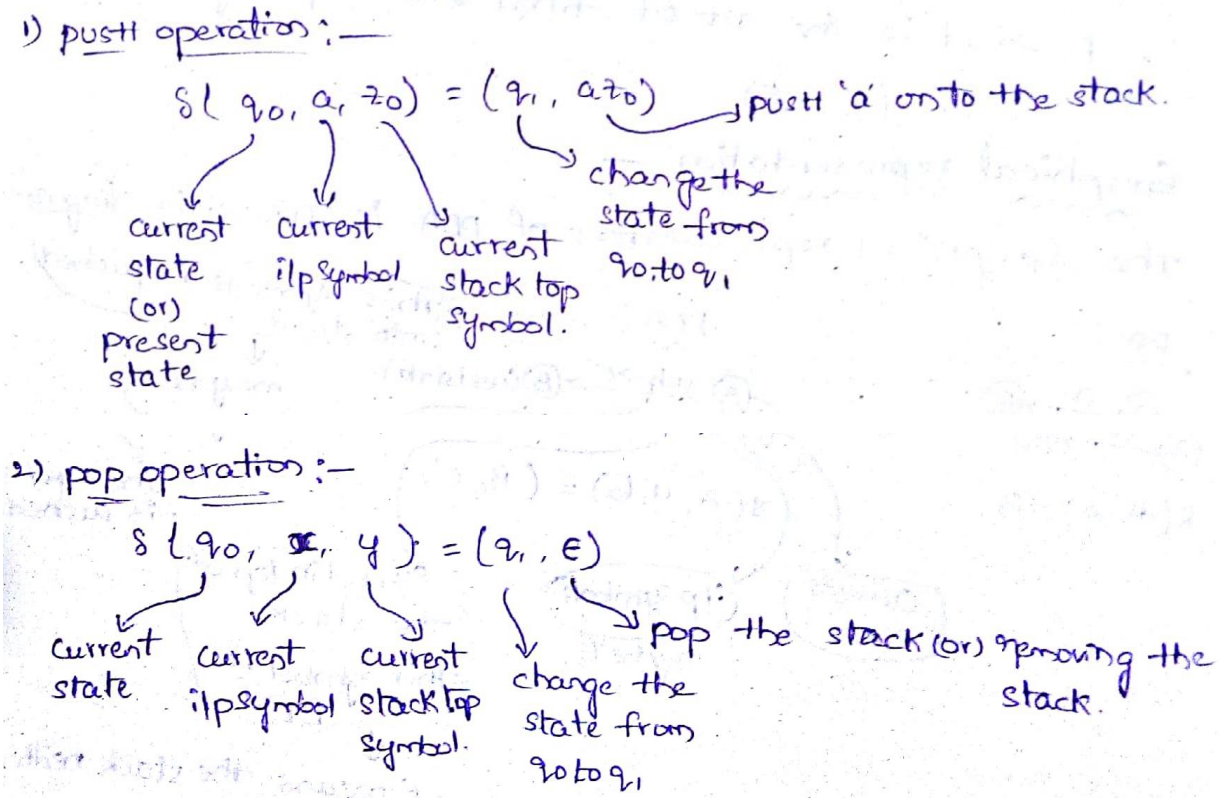
ID remembers the state and stack content

It was defined by triple, ID: (q,x,z) where 'q' is a state, 'x' is input symbols of string and ' z' is a string of stack symbols.



From the above figure, if we are reading the current input symbol 'a2' at current state 'S1' and current stack symbol 'Z_0', then after a move we will reach to state S2 and there will be some new symbol on the top of the stack.

This description can be represented as.



LANGUAGE ACCEPTANCE BY PDA:

A language can be accepted by a PDA using two approaches:

1. **Acceptance by final stare:** The PDA accepts its input by consuming it and finally it enters the final state.
2. **Acceptance b empty stack:** On reading the input string from the initial configuration for some PDA, the stack of PDA becomes empty.

Note: If acceptance is defined by empty stack then there is no meaning of final state and it is represented by **Ø**.

DESIGN OF PDA:

Example : consider a PDA $M = (\{q_0, q_1, q_2\}, \{a, c\}, \{a, Z_0\}, \delta, q_0, Z_0, \{q_2\})$ shown in below figure. Check the acceptability of string aacaa.

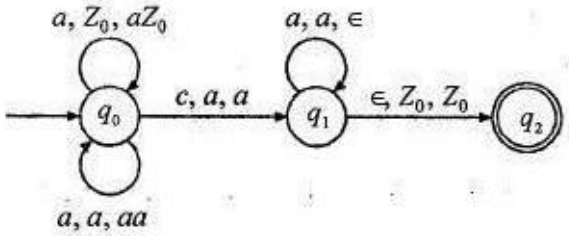


FIGURE : PDA accepting $\{a^n ca^n : n \geq 1\}$

Note : *Edges are labeled with Input symbol, stack symbol, written symbol on the stack.*

Solution :

The transition function δ is defined as follows :

$\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$,

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

$\delta(q_0, c, a) = \{(q_1, a)\}$,

$\delta(q_1, a, a) = \{(q_1, \in)\}$, and

$\delta(q_1, \in, Z_0) = \{(q_2, Z_0)\}$

Following moves are carried out in order to check acceptability of string *aacaa* :

$(q_0, aacaa, Z_0) \vdash (q_0, acaa, aZ_0)$

$\vdash (q_0, caa, aaZ_0)$

$\vdash (q_1, aa, aaZ_0)$

$\vdash (q_1, a, aZ_0)$

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

$\vdash (q_2, \in, Z_0)$

Hence, $(q_0, aacaa, Z_0) \vdash_M^* (q_2, \in, Z_0)$.

Therefore, the string **aacaa** is accepted by M.

Like FSA, there are two types of PDAs:

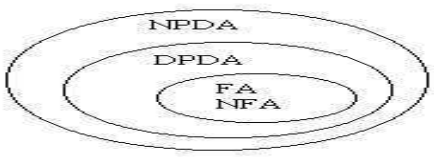
A deterministic PDA (DPDA) : is one in which every input string has a unique path through the machine.

: If all derivations in the design have to give only single move.

A nondeterministic PDA (NPDA) : is one in which we may have to choose among several paths for an input string.

:If derivation generates more than one move in the signing of a particular task.

If δ(q,a,A) contains more than one pair, then surely the PDA is non deterministic because we can choose among these pairs when deciding on the next move.



Non-deterministic Pushdown automata:

A non-deterministic pushdown automata is described by 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where

1. Q is finite and nonempty set of states,
2. Σ is input alphabet,
3. Γ is finite and nonempty set of pushdown symbols,
4. $q_0 \in Q$, is the starting state,
5. $Z_0 \in \Gamma$, is the starting (top most or initial)stack symbol, and
6. $F \subseteq Q$, is the set of final states.
7. δ, is the transition function,

Here , the transition function(δ) is given as

$T: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow P(Q \times \Gamma^*)$.$T: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma^*)$.

Ex: Consider the set of transition rules of an NPDA given by: δ(q1 , a, z) ={(q2 , cd), (q3 , ε)}

If at any time the control unit is in state q1, the input symbol read is 'a', and the symbol on the top of stack is 'z', then one of the following two cases can occur:
(a) The control unit tends to go into the state q2 and the string 'cd' replaces 'z' on top of the stack.
(b) The control unit goes into state q3 with the symbol z removed from the top of the stack.

In the deterministic case, when the function δ is applied, the automaton moves to a new state $q \in Q$ and pushes a new string of symbols $x \in \Gamma^*$ onto the stack. As we are dealing with non-deterministic pushdown automaton, the result of applying δ is a finite set of (q, x) pairs.

Ex: Consider following transitions.

$\delta(q0, a, 0) = \{(q1, 10), (q3, \varepsilon)\}$

δ (q0,ε , 0) = {(q3, ε)}

δ (q1, a, 1) = {(q1, 11)}

δ (q1, b, 1) = {(q2,ε)}

δ (q2, b, 1) = {(q2, ε)}

δ (q2,ε , 0) = {(q3,ε)}

It is possible for us to recognize the string "aaabbb" using the following sequence of "Moves":

(q0,aaabbb,0) ⊢ (q1,aabbb,10)

⊢ (q1,abbb,110)

⊢ (q1,bbb,1110)

⊢ (q2,bb,110)

⊢ (q2,b.10)

⊢ (q2,ε,0)

Equivalence of Pushdown Automata with Context-Free Grammar:

*CFG and PDA are equivalent in power: a CFG generates a context-free language and a PDA recognizes a context-free language.

A language is context-free iff some pushdown automaton recognizes it.

Note: This means that:

1. If a language 'L' is context-free then there is a PDA 'M' that recognizes it.
2. If a language is recognized by a PDA 'M' then there is a CFG 'G' that generates

Application: this equivalence allows a CFG to be used to specify a programming language and the equivalent PDA to be used to implement its compiler.

(A) CFG to PDA conversion:

Generally, left-most derivations is simulated by the PDA.

Let L=L(G), we construct PDA, M such that N(M)=L

M has:

Only one state {q}

Input symbols =Terminals of G

Stack symbols= All symbols of G

Start symbol=Start symbol of G

Algorithm to find PDA from CFG:

Let G=(V,T,P,S) and equivalent PDA, M=({q}, T,V∪T,δ,q0,S,Ø) // Ø represents accept by empty stack.

Step1: Convert the productions of CFG into GNF

Step2: PDA will have only one stare {q}

Step3: Start symbol of CFG will be start symbol of PDA

Step4: All non-terminals of CFG will be stack symbols of PDA and all terminals of CFG will be the input symbols of PDA.

Step5:

(i) If production of the form $A \rightarrow \beta$, Where $\beta \epsilon (VUT)^*$ and context-free rule says that LHS of production has single non-terminal , RHS of production has a string of both V and T.

Then make transition : $\delta(q,\varepsilon,A) \rightarrow (q,\beta)$ //replace A by β

(ii) If production of the form $A \rightarrow a$.

Then make transition: $\delta(q,a,a) \rightarrow (q,\varepsilon)$ //pop top most symbol from stack.

(B) PDA to CFG conversion:

If there is a PDA M=({q}, Σ, Γ,δ,q_0,Z_0,F)then their exists CFG 'G' which is accepted by PDA M

Let G be a CFG which is generated by PDA. The G can be defined as G=(V,T,P,S) where S is the start symbol and the set of non-terminals V={S,q,q',Z_0}

Now , we get set of production rules using the following algorithm.

Algorithm:-

Rule1: The start symbol production rule can be S→ [q,Z0,q'] where q-indicates present state, q'-indicates next state, Z0 is he stack symbol.

Rule2: If there exists a move of PDA as then the production rule can be written as δ(q,a,Z0)=(q',ε)

[q,Z0,q']->a

Rule3: If there exists a move of PDA as δ(q,a,Z0)=[q',Z1Z2Z3….Zn) then the production rule can be written as:[q,Z0,q']→a[q1,Z1,q2][q2,Z2,q3][q3,Z3,q4]….[qn-1,Zn,q]

Example: Consider a nondeterministic PDA $M_1 = (\{q_0\}, \{a,b\}, \{a,b,S\}, \delta, q_0, S, \phi)$ which accepts the language $L = \{a^n b^n : n \geq 1\}$ by empty store, where δ is defined as follows :

$\delta(q_0, \in, S) = \{(q_0, ab), (q_0, aSb)\}$ (Two possible moves),

$\delta(q_0, a, a) = \{(q_0, \in)\}$, and $\delta(q_0, b, b) = \{(q_0, \in)\}$

Construct an equivalent PDA M_2 which accepts L in final state and check whether string $w = aabb$ is accepted or not ?

Solution : Following moves are carried out by PDA M_1 in order to accept $w = aabb$:

$$(q_0, aabb, S) \vdash (q_0, aabb, aSb)$$
$$\vdash (q_0, abb, Sb)$$
$$\vdash (q_0, abb, abb)$$
$$\vdash (q_0, bb, bb)$$
$$\vdash (q_0, b, b)$$
$$\vdash (q_0, \in, \in)$$

Hence, $(q_0, aabb, S) \overset{*}{\underset{M_1}{\vdash}} (q_0, \in, \in)$

Therefore, $w = aabb$ is accepted by M_1.

Example: PDA for $D = \{0^k 1^k \mid k \geq 0\}$

1) Read 0s from input, push onto stack until read 1.
2) Read 1s from input, while popping 0s from stack.
3) Enter accept state if stack is empty. (note: acceptance only at end of input)

Example: PDA for $B = \{ww^{\mathcal{R}} \mid w \in \{0,1\}^*\}$

1) Read and push input symbols. Sample Input: w=011 $ww^{\mathcal{R}} =$

0	1	1	1	1	0

2) Non-deterministically either repeat or go to (2).
3) Read input symbols and pop stack symbols, compare.
   If ever ≠ then thread rejects.
4) Enter accept state if stack is empty. (do in "software")

Practice Questions:

1. Construct a PDA which accepts the language defined by the following grammar:

S ⟶ E + T / T

T ⟶T*F / F

F ⟶ (E) / a

Show the moves of the PDA for the string a+a*a.

2. a. Write the procedure to construct CFG equivalent to a given PDA

   b. Let M=({q0,q1}, {0,1}, {x,z0}, δ, q0, z0,¢) where δ is given by:

δ(q0,0,z0)=(q0,xz0)

δ(q1,1,x)=(q1,∈)

δ(q0,0,x)=(q0,xx)

δ(q1,∈,x)=(q1,∈)

δ(q0,1,x)=(q1,∈)

δ(q1,∈,z0)=(q1,∈)

Construct a CFG for the PDAM.

3. Construct a PDA that recognizes strings which contain equal number of 0's and 1's.

4. Construct a pushdown automaton that recognizes even length palindromes over an alphabet {0, 1}.

5. Design Pushdown Automata (PDA) for the following language:

L = {w|w is a palindrome over {a, b}*} and show the moves of PDA for string 'abaaba'.

6. Convert the given PDA P=({q,r},{a,b},{X,Z},δ,q,Z,q) to a context free grammar. The transition function is given by:

(i) (q,0,Z)={(q,XZ)}

(ii) (q,0,X)={(q,XX)}

(iii) (q,1,X)={(r,ϵ)}

(iv) (r,1,X)={(r,ϵ)}

(v) (r,,)={(r,ϵ)}

7. Construct pushdown automata (PDA) for the following language: L = {a^{n+1} b^n : n≥0}

Draw the transition diagram trace the string 'aaaabbb'.

8. Write the steps to convert the PDA which accepts the string by Final state to PDA which accepts the string by empty stack.

9. Convert the PDA P =({p, q}, {0, 1}, {X,Z}, δ, q, Z, p) to a CFG, if transition function is given by:

δ(q,1,Z)={(q,XZ)} δ(q,1,X)={(q,XX)} δ(w,£,X)={(q,£)}

δ(q,0,X)={(p,X)} δ(p,1,X)={(p,£)} δ(p,0,Z)={(q,Z)}

10. Construct a PDA that recognizes balanced parentheses.

11. Construct a PDA that recognizes strings of type $a^ib^jc^{i+j}$.