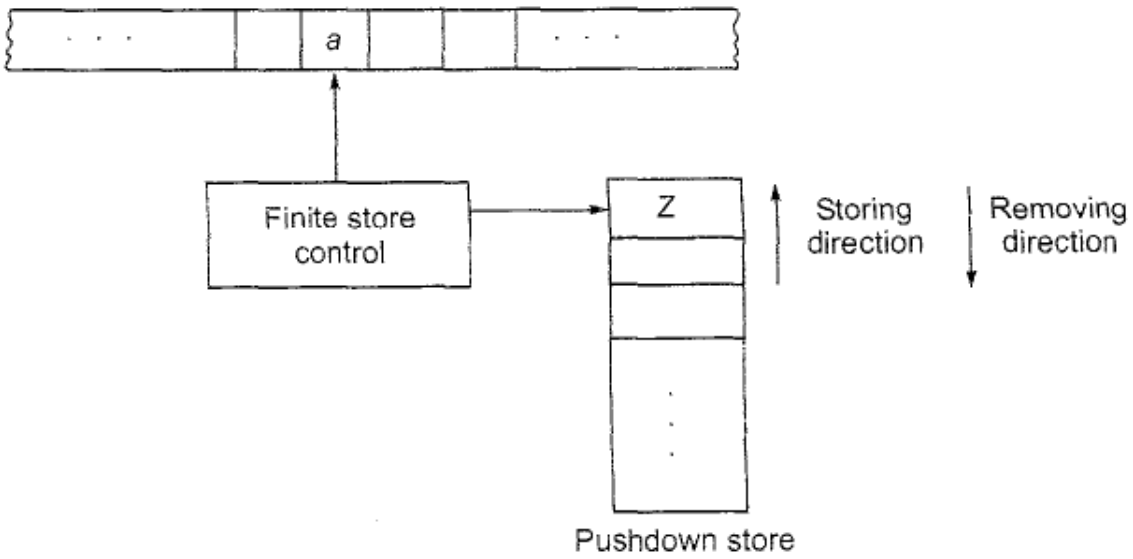
UNIT-3 (Part-1)
Pushdown Automata(PDA)

PDA

- Pushdown Automata is a way to implement a CFG in the same way we design DFA for a regular grammar. A DFA can remember a finite amount of information, but a PDA can remember an infinite amount of information.
- Pushdown Automata is simply an NFA augmented with an "external stack memory". The addition of stack is used to provide a last-in-first-out memory management capability to Pushdown automata. Pushdown automata can store an unbounded amount of information on the stack. It can access a limited amount of information on the stack. A PDA can push an element onto the top of the stack and pop off an element from the top of the stack. To read an element into the stack, the top elements must be popped off and are lost.
- A PDA is more powerful than FA. Any language which can be acceptable by FA can also be acceptable by PDA. PDA also accepts a class of language which even cannot be accepted by FA. Thus PDA is much more superior to FA.
- Pushdown automata, PDA, are a new type of computation model. PDAs are like NFAs but have an extra component called a stack .The stack provides additional memory beyond the finite amount available in the control .The stack allows PDA to recognize some nonregular languages.

Model of PDA



Input tape: The input tape is divided into squares (or cells), each square containing a single symbol from the input alphabet.
Reading head: The head examines only one square at a time and can move one square to the right side. Initially the head is places at the leftmost square of the tape.

Finite control: It can be in any of a finite number of states and change the state in some defined manner. Initially finite control is set to initial state(starting sate)
Stack: The stack is a structure in which we can push and remove the items from one end only. It has an infinite size. In PDA, the stack is used to store the items temporarily.

Formal definition of PDA:

The PDA can be defined as a collection of 7 components:

Q: the finite set of states

∑: the input set

Γ: a stack symbol which can be pushed and popped from the stack

$\mathbf{q}_0$ the initial state

Z: a start symbol which is in Γ.

F: a set of final states

δ: mapping function which is used for moving from current state to next state.

Instantaneous Description (ID)

ID is an informal notation of how a PDA computes an input string and make a decision that string is accepted or rejected.

An instantaneous description is a triple (q, w, α) where:

q describes the current state.

w describes the remaining input.

α describes the stack contents, top at the left.

Turnstile Notation:

⊢ sign describes the turnstile notation and represents one move.

⊢* sign describes a sequence of moves.

For example,

(p, b, T) ⊢ (q, w, α)

In the above example, while taking a transition from state p to q, the input symbol 'b' is consumed, and the top of the stack 'T' is represented by a new string α.

Example 1:

Design a PDA for accepting a language $\{a^n b^{2n} \mid n>=1\}$.

AD

Solution: In this language, n number of a's should be followed by 2n number of b's. Hence, we will apply a very simple logic, and that is if we read single 'a', we will push two a's onto the stack. As soon as we read 'b' then for every single 'b' only one 'a' should get popped from the stack.

The transition functions can be constructed as follows:

$\delta(q0, a, Z_0) = (q0, aaZ_0)$
$\delta(q0, a, a) = (q0, aaa)$

Now when we read b, we will change the state from q0 to q1 and start popping corresponding 'a'. Hence,

$\delta(q0, b, a) = (q1, \varepsilon)$

Thus this process of popping 'b' will be repeated unless all the symbols are read. Note that popping action occurs in state q1 only.

$\delta(q1, b, a) = (q1, \varepsilon)$

After reading all b's, all the corresponding a's should get popped. Hence when we read ε as input symbol then there should be nothing in the stack. Hence the move will be:

$\delta(q1, \varepsilon, Z_0) = (q2, \varepsilon)$

Where

PDA = ({q0, q1, q2}, {a, b}, {a, Z_0}, δ, q0, Z_0, {q2})

We can summarize the ID as:

$\delta(q0, a, Z_0) = (q0, aaZ_0)$
$\delta(q0, a, a) = (q0, aaa)$
$\delta(q0, b, a) = (q1, \varepsilon)$
$\delta(q1, b, a) = (q1, \varepsilon)$
$\delta(q1, \varepsilon, Z_0) = (q2, \varepsilon)$

Now we will simulate this PDA for the input string "aaabbbbbb".

$\delta(q0, aaabbbbbb, Z_0) \vdash \delta(q0, aabbbbbb, aaZ_0)$
$\vdash (q0, abbbbbb, aaaaZ_0)$
$\vdash (q0, bbbbbb, aaaaaaZ_0)$
$\vdash (q1, bbbbb, aaaaaZ_0)$
$\vdash (q1, bbbb, aaaaZ_0)$
$\vdash (q1, bbb, aaaZ_0)$
$\vdash (q1, bb, aaZ_0)$
$\vdash (q1, b, aZ_0)$
$\vdash (q1, \varepsilon, Z_0)$
$\vdash (q2, \varepsilon, \varepsilon)$
ACCEPT

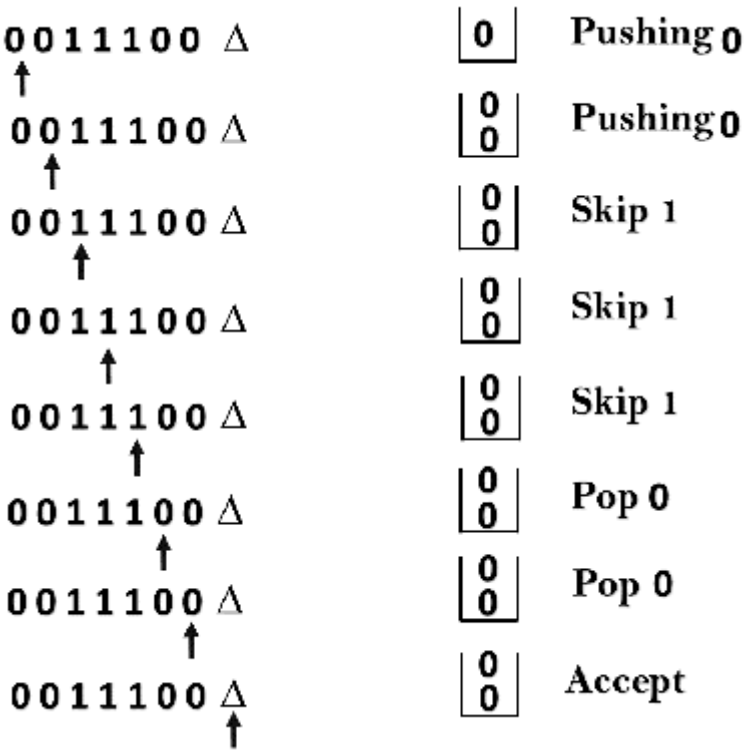
Example 2:

Design a PDA for accepting a language $\{0^n1^m0^n \mid m, n>=1\}$.

Solution: In this PDA, n number of 0's are followed by any number of 1's followed n number of 0's. Hence the logic for design of such PDA will be as follows:

Push all 0's onto the stack on encountering first 0's. Then if we read 1, just do nothing. Then read 0, and on each read of 0, pop one 0 from the stack.

For instance:



This scenario can be written in the ID form as:

$\delta(q0, 0, Z_0) = \delta(q0, 0Z_0)$
$\delta(q0, 0, 0) = \delta(q0, 00)$
$\delta(q0, 1, 0) = \delta(q1, 0)$
$\delta(q0, 1, 0) = \delta(q1, 0)$
$\delta(q1, 0, 0) = \delta(q1, \varepsilon)$
$\delta(q0, \varepsilon, Z_0) = \delta(q2, Z_0)$ (ACCEPT state)

Now we will simulate this PDA for the input string "0011100".

$(q0, 0011100, Z_0) \vdash (q0, 011100, 0Z_0)$
$\vdash (q0, 11100, 00Z_0)$
$\vdash (q0, 1100, 00Z_0)$
$\vdash (q1, 100, 00Z_0)$
$\vdash (q1, 00, 00Z_0)$
$\vdash (q1, 0, 0Z_0)$
$\vdash (q1, \varepsilon, Z_0)$
$\vdash (q2, \varepsilon, Z_0)$
ACCEPT

PDA Acceptance

A language can be accepted by Pushdown automata using two approaches:

1. Acceptance by Final State: The PDA is said to accept its input by the final state if it enters any final state in zero or more moves after reading the entire input.

Let P =(Q, ∑, Γ, δ, q0, Z_0, F) be a PDA. The language acceptable by the final state can be defined as:

$L(P) = \{w \mid (q0, w, Z_0) \vdash^* (p, \varepsilon, Z_0), p \in F\}$

2. Acceptance by Empty Stack: On reading the input string from the initial configuration for some PDA, the stack of PDA gets empty.

Let P =(Q, ∑, Γ, δ, q0, Z_0, F) be a PDA. The language acceptable by empty stack can be defined as:

$N(P) = \{w \mid (q0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon), p \in Q\}$

Equivalence of Acceptance by Final State and Empty Stack

- If L = N(P1) for some PDA P1, then there is a PDA P2 such that L = L(P2). That means the language accepted by empty stack PDA will also be accepted by final state PDA.
- If there is a language L = L (P1) for some PDA P1 then there is a PDA P2 such that L = N(P2). That means language accepted by final state PDA is also acceptable by empty stack PDA.

Example:

Construct a PDA that accepts the language L over {0, 1} by empty stack which accepts all the string of 0's and 1's in which a number of 0's are twice of number of 1's.

Solution:

There are two parts for designing this PDA:

- If 1 comes before any 0's
- If 0 comes before any 1's.

We are going to design the first part i.e. 1 comes before 0's. The logic is that read single 1 and push two 1's onto the stack. Thereafter on reading two 0's, POP two 1's from the stack. The δ can be

$\delta(q0, 1, Z_0) = (q0, 11, Z_0)$ Here Z_0 represents that stack is empty
$\delta(q0, 0, 1) = (q0, \varepsilon)$

Now, consider the second part i.e. if 0 comes before 1's. The logic is that read first 0, push it onto the stack and change state from q0 to q1. [Note that state q1 indicates that first 0 is read and still second 0 has yet to read].

Being in q1, if 1 is encountered then POP 0. Being in q1, if 0 is read then simply read that second 0 and move ahead. The δ will be:

$\delta(q0, 0, Z_0) = (q1, 0Z_0)$
$\delta(q1, 0, 0) = (q1, 0)$
$\delta(q1, 0, Z_0) = (q0, \varepsilon)$ (indicate that one 0 and one 1 is already read, so simply read the second 0)
$\delta(q1, 1, 0) = (q1, \varepsilon)$

Now, summarize the complete PDA for given L is:

$\delta(q0, 1, Z_0) = (q0, 11Z_0)$
$\delta(q0, 0, 1) = (q1, \varepsilon)$
$\delta(q0, 0, Z_0) = (q1, 0Z_0)$
$\delta(q1, 0, 0) = (q1, 0)$
$\delta(q1, 0, Z_0) = (q0, \varepsilon)$
$\delta(q0, \varepsilon, Z_0) = (q0, \varepsilon)$ ACCEPT state

Non-deterministic Pushdown Automata

The non-deterministic pushdown automata is very much similar to NFA. We will discuss some CFGs which accepts NPDA.

The CFG which accepts deterministic PDA accepts non-deterministic PDAs as well. Similarly, there are some CFGs which can be accepted only by NPDA and not by DPDA. Thus NPDA is more powerful than DPDA.

Example:

Design PDA for Palindrome strips.

Solution:
Suppose the language consists of string L = {aba, aa, bb, bab, bbabb, aabaa, ...]. The string can be odd palindrome or even palindrome. The logic for constructing PDA is that we will push a symbol onto the stack till half of the string then we will read each symbol and then perform the pop operation. We will compare to see whether the symbol which is popped is similar to the symbol which is read. Whether we reach to end of the input, we expect the stack to be empty.

This PDA is a non-deterministic PDA because finding the mid for the given string and reading the string from left and matching it with from right (reverse) direction leads to non-deterministic moves. Here is the ID.

1. $\delta(q1, a, Z) = (q1, aZ)$
2. $\delta(q0, b, Z) = (q1, bZ)$
3. $\delta(q0, a, a) = (q1, aa)$
4. $\delta(q1, a, b) = (q1, ab)$
5. $\delta(q1, a, b) = (q1, ba)$
6. $\delta(q1, b, b) = (q1, bb)$
7. $\delta(q1, a, a) = (q2, \varepsilon)$
8. $\delta(q1, b, b) = (q2, \varepsilon)$
9. $\delta(q2, a, a) = (q2, \varepsilon)$
10. $\delta(q2, b, b) = (q2, \varepsilon)$
11. $\delta(q2, \varepsilon, Z) = (q2, \varepsilon)$

Pushing the symbols onto the stack

Popping the symbols on reading the same kind of symbol

Simulation of abaaba

(q1, abaaba, Z)	Apply rule 1
⊢ (q1, baaba, aZ)	Apply rule 5
⊢ (q1, aaba, baZ)	Apply rule 4
⊢ (q1, aba, abaZ)	Apply rule 7
⊢ (q2, ba, baZ)	Apply rule 8
⊢ (q2, a, aZ)	Apply rule 7
⊢ (q2, ε, Z)	Apply rule 11

$\vdash (q2, \varepsilon, \varepsilon)$ Accept

PDA for L ={$a^n b^n c^m$ / n,m≥1} by final state

P=({q0 ,q1,q2,qf},{a,b,c},δ,{a,Z_0},,q0,Z_0,{qf})

Where δ defined by:

$\delta(q0, a, Z_0) = (q0, aZ_0)$

$\delta(q0, a, a) = (q0, aa)$

$\delta(q0, b, a) = (q1, \varepsilon)$

$\delta(q1, b, a) = (q1, \varepsilon)$

$\delta(q1, c, Z_0) = (q2, Z_0)$

$\delta(q2, c, Z_0) = (q2, Z_0)$

$\delta(q2, \varepsilon, Z_0) = (qf, Z_0)$

PDA for L= {$a^n b^n c^m d^m$ / n,m≥1} by final state.

P=({q0 ,q1,q2,q3,qf},{a,b,c,d},δ,{a,c,Z_0},q0,Z_0,{qf})

Where δ defined by:

$\delta(q0, a, Z_0) = (q0, aZ_0)$

$\delta(q0, a, a) = (q0, aa)$

$\delta(q0, b, a) = (q1, \varepsilon)$

$\delta(q1, b, a) = (q1, \varepsilon)$

$\delta(q1, c, Z_0) = (q2, cZ_0)$

$\delta(q2, c, c) = (q2, cc)$

$\delta(q2, d, c) = (q3, \varepsilon)$

$\delta(q3, d, c) = (q3, \varepsilon)$

$\delta(q3, \varepsilon, Z_0) = (qf, Z_0)$

PDA for L= { ww^R / w ∈ $(a,b)^*$ }

P=({q0 ,q1,qf},{a,b},δ,{a,b,Z_0},q0,Z_0,{qf})

Where δ defined by:

$\delta(q0, a, Z_0) = (q0, aZ_0)$
$\delta(q0, a, a) = (q0, aa)$
$\delta(q0, b, Z_0) = (q0, bZ_0)$
$\delta(q0, b, b) = (q0, bb)$
$\delta(q0, a, b) = (q0, ab)$
$\delta(q0, b, a) = (q0, ba)$

// this is decision step
$\delta(q0, a, a) = (q1, \varepsilon)$
$\delta(q0, b, b) = (q1, \varepsilon)$

$\delta(q1, b, b) = (q1, \varepsilon)$
$\delta(q1, a, a) = (q1, \varepsilon)$

$\delta(q1, \varepsilon, Z_0) = (qf, Z_0)$

PDA for $L = \{ a^n b^m c^n \ / \ n,m \geq 1\}$ by empty stack

$P=(\{q0 ,q1,q2\},\{a,b,c\},\delta,\{a,Z_0\},q0,Z_0,\phi)$

Where δ defined by:

$\delta(q0, a, Z_0) = (q0, aZ_0)$
$\delta(q0, a, a) = (q0, aa)$

$\delta(q0, b, a) = (q1, a)$
$\delta(q1, b, a) = (q1, a)$

$\delta(q1, c, a) = (q2, \varepsilon)$
$\delta(q2, c, a) = (q2, \varepsilon)$

$\delta(q2, \varepsilon, Z_0) = (q2, \varepsilon)$

PDA for $L= \{a^n b^{2n} \ / \ n \geq 1\}$

$P=(\{q0 ,q1,qf\},\{a,b\},\delta,\{a,Z_0\},q0,Z_0,\{qf\})$

Where δ defined by:

$\delta(q0, a, Z_0) = (q0, aaZ_0)$

$\delta(q0, a, a) = (q0, aaa)$

$\delta(q0, b, a) = (q1, \varepsilon)$

$\delta(q1, b, a) = (q1, \varepsilon)$

$$\delta(q1, \varepsilon, Z_0) = (qf, Z_0)$$

PDA for number of a(w) = number of b(w)

P=({q0,qf},{a,b},δ,{a,b,Z_0},q0,Z_0,{qf})

Where δ defined by:

$$\delta(q0, a, Z_0) = (q0, aZ_0)$$

$$\delta(q0, a, a) = (q0, aa)$$

$$\delta(q0, b, Z_0) = (q0, bZ_0)$$

$$\delta(q0, b, b) = (q0, bb)$$

$$\delta(q0, b, a) = (q0, \varepsilon)$$

$$\delta(q0, a, b) = (q0, \varepsilon)$$

$$\delta(q0, \varepsilon, Z_0) = (qf, Z_0)$$

Difference between NPDA and DPDA:

S. No	**DPDA(Deterministic Pushdown Automata)**	**NDPA(Non-deterministic Pushdown Automata)**
1.	It is less powerful than NPDA.	It is more powerful than DPDA.
2.	It is possible to convert every DPDA to a corresponding NPDA.	It is not possible to convert every NPDA to a corresponding DPDA.
3.	The language accepted by DPDA is a subset of the language accepted by NDPA.	The language accepted by NPDA is not a subset of the language accepted by DPDA.
4.	The language accepted by DPDA is called DCFL (Deterministic Context-free Language) which is a subset of NCFL (Non-deterministic Context-free Language) accepted by NPDA.	The language accepted by NPDA is called NCFL (Non-deterministic Context-free Language).

Tutorial Questions

1. Construct a PDA for L={$a^n b^n$ /n ≥1}.Draw transition diagram. Using the instantaneous description notation process the string **aaabbb**
2. Define PDA and instantaneous description of PDA. Obtain a PDA to accept the language L={ wcw^R : w ∈{a,b}*}.Draw transition diagram of PDA. Show the moves by this PDA for string **abbcbba**
3. Explain the various ways of determining the acceptability of Pushdown Automata.
4. Construct a PDA that accepts L = {$0^n 1^n$ | n ≥0}
5. Design a PDA to accept the language of balanced parenthesis.
6. Design PDA for the following languages by Empty Stack?
   i) L = { $a^n b^n c^m$ /n, m>=1}
   ii) L = { $a^n b^m c^n$ /n, m>=1}
   iii) L = { $a^m b^n c^n$ /n, m>=1}
7. Design PDA for the following languages by Empty Stack?
   i) L = { $a^m b^n c^{m+n}$ /n, m>=1}
   ii) L = { $a^m b^{m+n} c^n$ /n, m>=1}
   iii) L = { $a^{m+n} b^m c^n$ /n, m>=1}
8. Design PDA for the following languages by Empty Stack?
   i) L = { $a^n b^m c^m d^n$ /n, m>=1}
   ii) L = { $a^n b^n c^m d^m$ /n, m>=1}
9. Design PDA for the following languages by final state?
   i) L = { $a^n bc^n$ /n>=1}
   ii) L = { $ab^n c^n$ /n>=1}
   iii) L = { $a^n b^n c$ /n>=1}
10. Design PDA for the following languages by final state?
   i) L = { $a^n b^3 c^n$ /n>=1}
   ii) L = { $a^3 b^n c^n$ /n>=1}
   iii) L = { $a^n b^n c^3$ /n>=1}
11. Design PDA for the following languages by final state?
   i) L = { $a^n bc^{2n}$ /n>=1}
   ii) L = { $ab^n c^{2n}$ /n>=1}
   iii) L = { $a^n b^{2n} c$ /n>=1}
12. Design PDA for the following languages by final state?
   i) L = { $a^n b^{n+1}$ /n>=1}
   ii) L = { $a^n b^{2n+1}$ /n>=1}
   iii) L = { $a^{2n} b^{3n}$/n>=1}
   iv) L = { $a^{3n} b^{2n}$/n>=1}
13. Design PDA for the following languages by final state?
   i) L = { $a^m b^n$ /n, m>0,m ≠ n}

ii) $L = \{ a^m b^n / n, m>0, m>n\}$

iii) $L = \{ a^m b^n / n, m>0, m<n\}$

14. Construct a PDA for $L=\{ wcw^R / w \in (0+1)^*\}$
15. Design a non deterministic push down automata for the following languages **$L1=\{a^n b^n | n>=0\}$, $L2=\{ww^R | w \text{ in } (0+1)^*\}$**
16. Describe the components of Push Down Automata.
17. Design a PDA for accepting a language $\{a^n b^{2n} | n>=1\}$.Show the moves of the PDA for the string **aabbbb**
18. Design a PDA for accepting a language $\{a^{2n} b^n | n>=1\}$.Show the moves of the PDA for the string **aaaabb**
19. Differentiate between Deterministic PDA and Non-deterministic PDA.
    (or) Distinguish between a DPDA and NPDA
20. Define Push Down Automata. Explain the basic structure of PDA with a neat graphical representation.
21. Construct a PDA for recognizing the language $L = \{a^i b^j c^k / i,j,k \in N, i+k=j\}$.
22. Construct a PDA for recognizing the language of all strings over the input alphabet {a,b} such that the number of b's in each string are equal the number of a's. Show the moves of the PDA for the string **ababbbaa**
23. Construct a PDA for recognizing the language of all strings over the input alphabet {a,b} such that the number of b's in each string are twice the number of a's. Show the moves of the PDA for the string **abbabbbba**
24. Design PDA for recognizing the language of palindromes over the alphabet {0,1}.Draw the computations tree showing all possible moves for the strings **00100** and **00101**
25. When do you say that a language is recognized or accepted by a PDA? Design a PDA for $L=\{a^i b^j c^k / j>=i+k$ and $i,j,k>0.\}$ Process the string **aaabbbbbbcc** using instantaneous description.
26. Design PDA for recognizing the language $L=\{a^i b^j / j<=i$ and $i,j>0\}$ Show the moves of the PDA for the string **aaabb**
27. Does push down automata have memory? Justify your answer. Mention the applications of PDA.