Unit - IV

Pushdown Automata

1. Define Pushdown Automata. Explain the basic structure of PDA with a neat graphical representation.

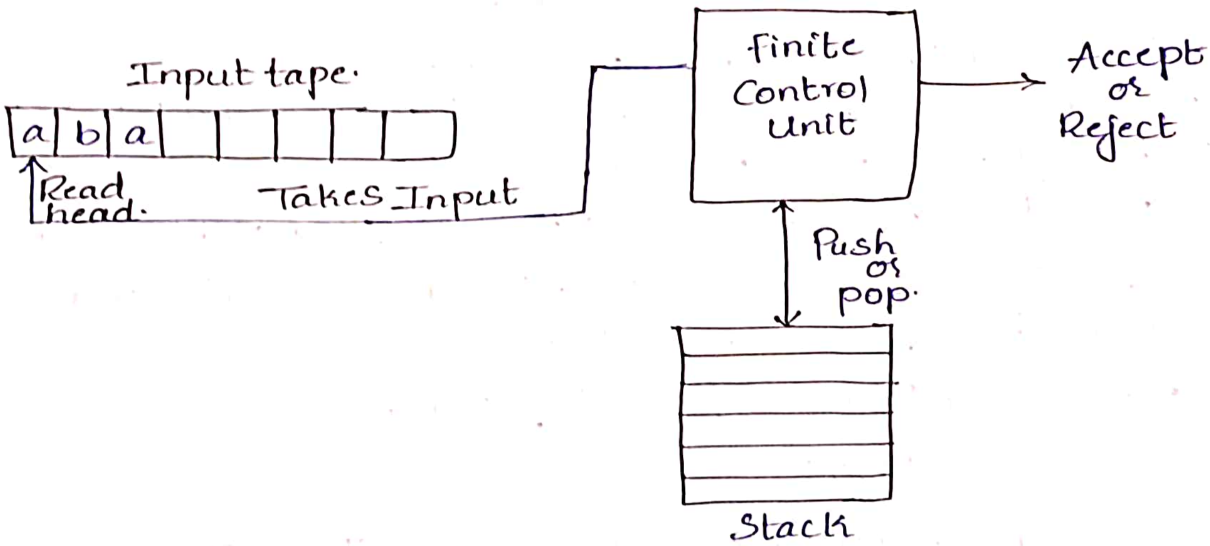
Pushdown Automata: A pushdown Automata (PDA) is a way to implement a Context Free Grammar in a Similar way we design finite Automata for Regular Grammar.

→ It is more powerful than FSM

→ FSM has a very limited memory but PDA has more memory.

→ PDA = Finite State Machine + A stack.

Structural Components of PDA:



1. The PDA Contains 3 parts.
   1. Input tape
   2. Finite State Control
   3. Stack.

⇒ Input tape is divided into no. of Cells where each Cell (It h
Store 1 input Symbol at a time. It

⇒ finite State Control Contains finite Stack Set of States, it maintains two read heads.

⇒ 1 read head points to the 1st Symbol of a input tape after reading 1st Symbol the read head moves to forward directio

⇒ A FA having Only input tape and finite State Control Unit PD in addition to the input tape finite State Control are need Stack also.

⇒ Stack is a data structure and it having a finite amount of memory.

⇒ Stack performs two Operations 1. push
2. pop

⇒ Stack is Used to store the input Symbols.

⇒ push operation push the Symbol to the Stack and pop operation Pop the Symbol from the Stack.

⇒ Stack provides two States i.e, either accepted or Rejected.

2) Describe the Components of pushDown Automata and Mention the applications of PDA.

Components of PDA: Refer 1st question.

Applications of PDA:

1. PDA is Used for deriving a String from the grammar.
2. PDA is Used for designing Top-down Parser and bottom-up Parser in Compiler design.
3. It works on regular grammar and Context-Free Grammar
4. It accepts regular language and CFL.

It has remembering Capability by maintaining a stack.
It is more powerful than Finite Automata.

) Design a PDA for accepting a language $\{a^n b^{2n} [n>=1]\}$

o): Given $L = \{a^n b^{2n} \mid n>=1\}$

$L = \{abb, aabbbb, aaabbbbbb ------\}$

Procedure:

Step1: Whenever 'a' occurs, for every occurrence of 'a', push two 'a's on to the stack.

Step2: Whenever b- Occurs Change the State Only for 1^{st} Operation and pop a from the Stack.

Step3: Repeat Step2 Until Stack is Empty.

It has 7 tuples.

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ ⟹ Let's Consider the input String abb.

$\delta(q_0, a, z_0) = (q_0, az_0)$ // push 'a'

$\delta(q_0, a, a) = (q_0, aaz_0)$ // push 'a'

$\delta(q_0, a, a) = (q_0, aaaz_0)$ // push 'a'

$\delta(q_0, b, a) = (q_1, \epsilon)$ // Change the State $q_0 \to q_1$ for first b

$\delta(q_1, b, a) = (q_1, \epsilon)$ // pop 'a'

$\delta(q_1, b, a) = (q_1, \epsilon)$ // pop 'a'.

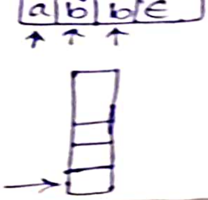
$\delta(q_1, \epsilon, z_0) = (q_2, \epsilon)$

$\delta(q_0, a, z_0) = (q_0, aaz_0)$ // push aa

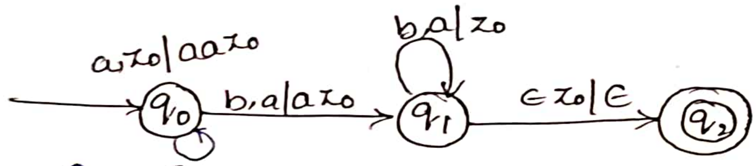
$\delta(q_0, b, a) = (q_1, az_0)$ // pop 'a'

$\delta(q_1, b, a) = (q_1, \epsilon)$ // pop 'a'

$\delta(q_1, \epsilon, z_0) = (q_2, \epsilon)$



Transition Diagram:



$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$T = \{a, z_0\}$

$q_0 = q_0$

$z_0 = z_0$

$F = \{q_2\}$

4. Develop a PDA that accepts the strings of the form $a^n b^{2n}$ Where $n>1$.

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

$L = \{aabbbb, aaabbbbbb, \dots\}$

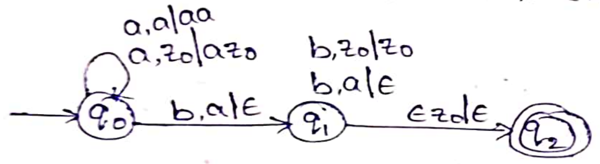
Step 1: For every occurrence of a, push 'a' on to the stack.

Step 2: For ~~ev~~first occurrence of b, change the state and

(i) If input is b and top of stack 'a' then perform pop operation.

(ii) If input is b and top of the stack is z_0 then perform unchange operation.

Step 3: Repeat step 2 until stack is empty.



let us consider the input string aabbbb

$\delta(q_0, aabbbb, z_0) \vdash (q_0, abbbb, az_0) \rightarrow$

$\vdash (q_0, bbbb, aaz_0) \rightarrow$

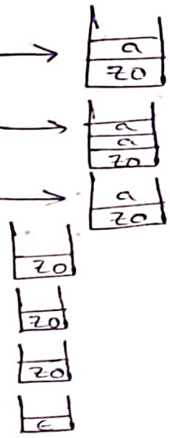
$\vdash (q_\phi, bbb, az_0) \rightarrow$

$\vdash (q_\phi, bb, z_0) \rightarrow$

$\vdash (q_\phi, b, z_0) \rightarrow$

$\vdash (q_\phi, \epsilon, z_0) \rightarrow$

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



5) Construct the PDA for the given grammar $S \rightarrow AA|a, A \rightarrow SA|b$

Sol: Given CFG is $S \rightarrow AA|a$

$A \rightarrow SA|b$

The CFG Contains 4 tuples

$G = (V, T, P, S)$

$= (\{S, A\}, \{a, b\}, P, S)$

The P.D.A Contains 7 Tuples.

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

$= (\{S, A\}, \{a, b\}, \{S, A, a, b\}, \delta, q_0, z_0, F)$

Step1: The given CFG is in GNF.

Step2: For Non-terminal:

$S \rightarrow AA \Rightarrow \delta(q_0, \epsilon, S) = (q_0, AA)$

$S \rightarrow a \Rightarrow \delta(q_0, \epsilon, S) = (q_0, a)$

$A \rightarrow SA \Rightarrow \delta(q_0, \epsilon, A) = (q_0, SA)$

$A \rightarrow b \Rightarrow \delta(q_0, \epsilon, A) = (q_0, b)$

Step3:

For - terminals:

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

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

i/p String : abab

$\delta(q_0, abab, S) \vdash (q_0, abab, AA)$

$\vdash (q_0, abab, SAA)$

$\vdash (q_0, \cancel{a}bab, \cancel{a}AA)$

$\vdash (q_0, bab, AA)$

$\vdash (q_0, \cancel{b}ab, \cancel{b}A)$

$\vdash (q_0, ab, SA)$

$\vdash (q_0, \cancel{a}b, \cancel{a}A)$

$\vdash (q_0, b, A)$

$\vdash (q_0, \cancel{b}, \cancel{b})$

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

6) Construct a PDA for the following grammar $S \to 0B0 | 1B1$, $B \to 0B | 11 | 2$.

Given CFG is $S \to 0B0 | 1B1$

$B \to 0B | 11 | 2$.

CFG Contains 4 tuples

$G = (V, T, P, S)$

$= (\{B, S\}, \{0, 1, 2\}, P, S)$.

PDA Contains 7 tuples.

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

$= (\{S, B\}, \{0, 1, 2\}, \{S, B, 0, 1, 2\}, \delta, q_0, z_0, F)$.

For Non-terminals:

$S \to 0B0 \Rightarrow \delta(q_0, \epsilon, S) = (q_0, 0B0)$

$S \to 1B1 \Rightarrow \delta(q_0, \epsilon, S) = (q_0, 1B1)$

$B \to 0B \Rightarrow \delta(q_0, \epsilon, B) = (q_0, 0B)$

$B \to 11 \Rightarrow \delta(q_0, \epsilon, B) = (q_0, 11)$

$B \to 2 \Rightarrow \delta(q_0, \epsilon, B) = (q_0, 2)$

For terminals:

$\delta(q_0, 0, 0) = (q_0, \epsilon)$

$\delta(q_0, 1, 1) = (q_0, \epsilon)$

$\delta(q_0, 2, 2) = (q_0, \epsilon)$

i/p string: 0101110

$$
\begin{aligned}
\delta(q_0, 0101110, S) &\vdash (q_0, \cancel{0}101110, \cancel{0}B0) \\
&\vdash (q_0, 101110, B0) \\
&\vdash (q_0, \cancel{1}01110, \cancel{1}B10) \\
&\vdash (q_0, 01110, B10) \\
&\vdash (q_0, \cancel{0}1110, \cancel{0}B10) \\
&\vdash (q_0, 1110, B10) \\
&\vdash (q_0, \cancel{1}110, \cancel{1}110) \\
&\vdash (q_0, \cancel{1}10, \cancel{1}10) \\
&\vdash (q_0, 10, 10) \\
&\vdash (q_0, 0, 0) \\
&\vdash (q_0, \epsilon, \epsilon)
\end{aligned}
$$

7. Discuss Various Steps for Converting CFG to PDA with your own Example.

Conversion of CFG to PDA:

To Convert CFG to PDA we have follow below procedure.

Step1: Convert the given CFG into G.N.F

Step2: For Non-terminal i.e, $A \rightarrow \alpha$ then $\alpha \in (V \cup T)^*$

Write T.F is $\delta(q_0, \epsilon, A) = \delta(q_0, \alpha)$

Step3: For terminal, Suppose 'a'

$\delta(q_0, a, a) = (q_0, \epsilon)$ // POP

Suppose +

$\delta(q_0, +, +) = (q_0, \epsilon)$ # POP

8. For example: Refer 5th question.

8) Is a push-down automation with two stacks equivalent to turning machine? Justify your answer with proper explanation.

No, a two stack PDA is not equivalent to a Turning machine. Here's why

1. Expressive power:

A Turning Machine (TM) is more powerful than a two Stac PDA in terms of Computational Capability. TMS Can recognize languages that are not Context-free, Whereas PDAs, even with two stacks, are limited to recognizing only Context-free languages.

2. Memory:

While a Turning Machine has an infinite tape as its memory, a two Stack PDA has limited memory in the form of two stacks. This limitation restricts the types of language that Can be recognised for Computed.

3. Computation Model:

Turning Machines Can Simulate any algorithmic process Simulate Effectively, making them Capable of Solving a broader range of problems Compared to PDA's Which are more Suited for recognizing Certain types of languages.

4. Decidability:

Turning Machines Can decide recursively enumerable languages, Whereas PDAs Can Only decide Context-free languages.

⇒ While a two-stack PDA is a powerful Computational model Capable to recognizing Context-free languages, it is not equivalent to a Turning Machine due to differences in Expressive power, memory, Computation model and decidability.

9. Construct a PDA from the following CFG. $G = (\{S, X\}, \{a, b\}, P, S)$ where the productions are given below.

$S \to XS \mid \epsilon$

$A \to aXb \mid Ab \mid ab$

Given CFG is $S \to XS \mid \epsilon$

$A \to aXb \mid Ab \mid ab$

CFG Contains 4-tuples

$G = (V, T, P, S)$

$= (\{A, S\}, \{a, b\}, P, S)$

PDA Contains 7-tuples.

$M = (Q, \Sigma, T, \delta, q_0, z_0, F)$

$(\{A, S\}, \{a, b\}, \{A, S, a, b\}, \delta, q_0, z_0, F)$

For Non-terminals :

$S \rightarrow XS \Rightarrow \delta(q_0, \epsilon, S) = (q_0, XS)$

$A \rightarrow axb \Rightarrow \delta(q_0, \epsilon, A) = (q_0, axb)$

$A \rightarrow Ab \Rightarrow \delta(q_0, \epsilon, A) = (q_0, Ab)$

$A \rightarrow ab \Rightarrow \delta(q_0, \epsilon, A) = (q_0, ab)$

For terminals :

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

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

i/p string :

Discuss the notation and applications of two stack push down automata.

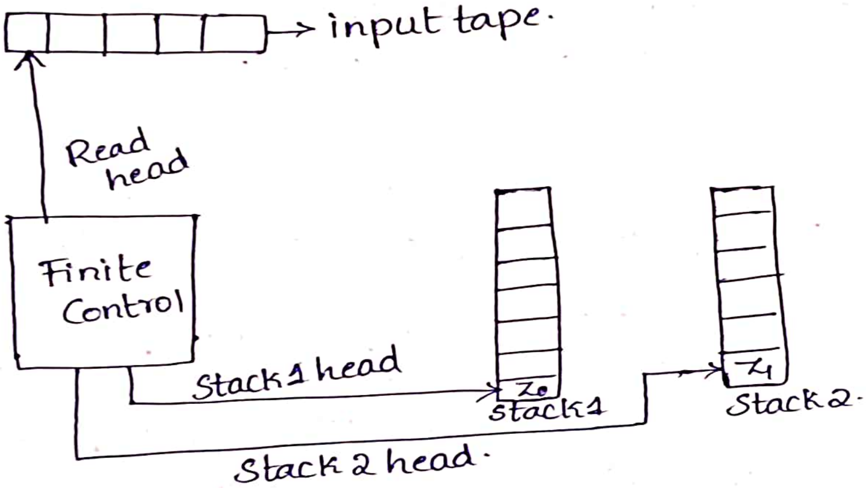
Two Stack PDA:

⇒ To increase the power of a PDA we can add one more stack.

⇒ A language which is accepted by two stack PDA which in also accepted by Turning Machine.

⇒ A Two Stack PDA contains mainly 3 parts.

1. Input tape
2. Finite Control
3. Two stacks.



⇒ Input tape is divided into no. of cells where each cell is store one symbol at a time.

⇒ finite Control represents the Current state of a machine.

⇒ It having two read heads, one read head points to the first symbol of a input tape. and Second read head points to the top most symbols of a stacks.

⇒ In Order to define two stack PDA by using 9 Tuples.

$$M = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, z_0, z_1, q_0, F)$$

$$M = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, z_0, z_1, q_0, F)$$

Where Q = Finite Set of States

Σ = input alphabet

Γ_1 = Stack 1 Symbols

Γ_2 = Stack 2 Symbols.

q_0 = Initial State

z_0 = top Symbol of Stack

z_1 = top Symbol of Stack

F = Set of final States

δ = Transition function.

$$\delta = Q \times \Sigma \cup \epsilon \times \Gamma_1 \times \Gamma_2 \rightarrow Q \times \Gamma_1^* \times \Gamma_2^*.$$

Applications:

1. Compiler Design: Two stack PDAs are used in the lexical analysis phase of Compilers to implement lexical analyzers or Scanners.
2. Parsing: Two Stack PDAs are Employed in parsing algorithms Such as LR(1) and LALR(1) parsing.
3. Natural Language processing (NLP): In NLP, two Stack PDAs Can be Utilized for Syntactic analysis of Sentences based on Context-free grammars.
4. Syntax Analysis in programming Languages: Two Stack PDAs are Used to perform Syntax analysis in programming languages.
5. XML processing: Two Stack PDAs Can be Employed in XML processing to Validate XML documents aganist Document Type.