PushDown Automata (PDA)

A pushdown automaton is a way to implement a context-free grammar in a similar 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.

Basically a pushdown automaton is −

"Finite state machine" + "a stack"

A pushdown automaton has three components −

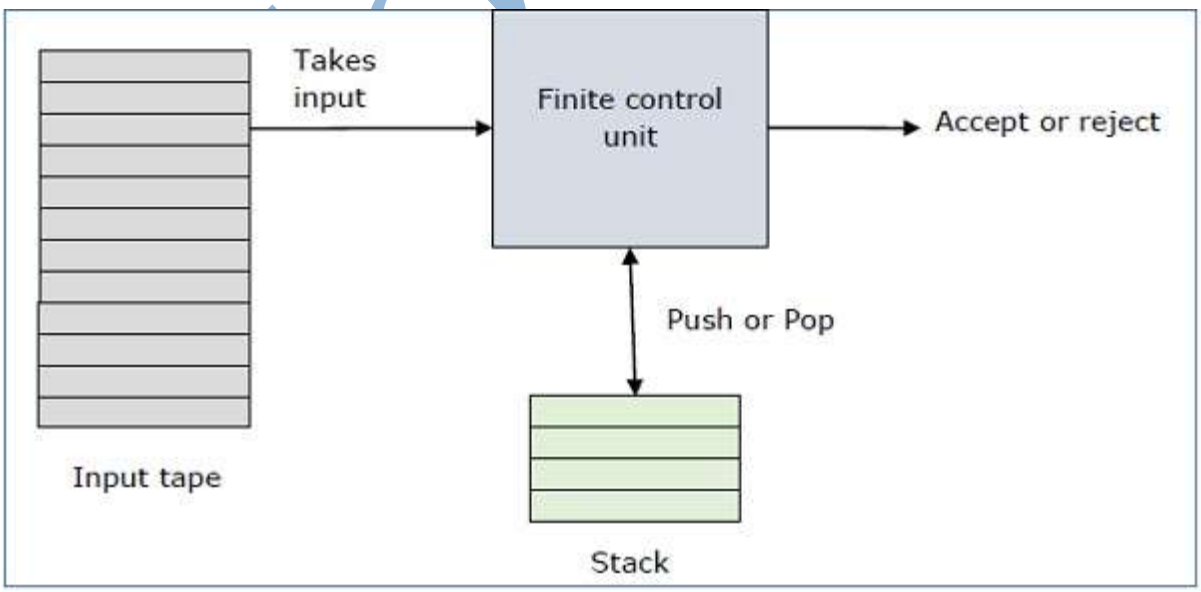
- an input tape,
- a control unit, and
- a stack with infinite size.

The stack head scans the top symbol of the stack.

A stack does two operations −

- **Push** − a new symbol is added at the top.
- **Pop** − the top symbol is read and removed.

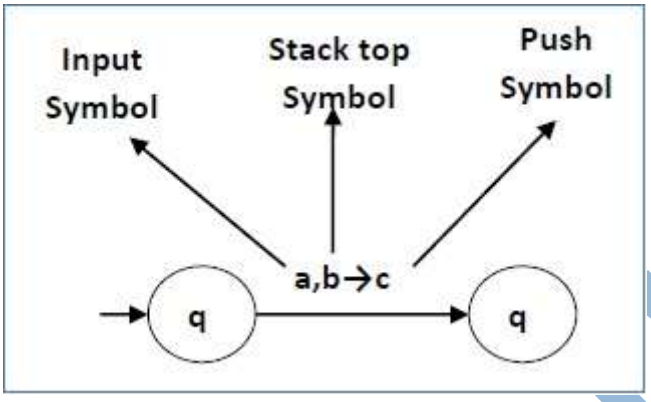
A PDA may or may not read an input symbol, but it has to read the top of the stack in every transition.



A PDA can be formally described as a 7-tuple $(Q, \sum, S, \delta, q_0, I, F)$ −

- **Q** is the finite number of states
- ∑ is input alphabet
- **S** is stack symbols
- **δ** is the transition function: Q × (∑ ∪ {ε}) × S × Q × S*
- **q_0** is the initial state (q_0 ∈ Q)
- **I** is the initial stack top symbol (I ∈ S)
- **F** is a set of accepting states (F ∈ Q)

The following diagram shows a transition in a PDA from a state q_1 to state q_2, labeled as a,b → c −



This means at state **q_1**, if we encounter an input string **'a'** and top symbol of the stack is **'b'**, then we pop **'b'**, push **'c'** on top of the stack and move to state **q_2**.

Terminologies Related to PDA

Instantaneous Description

The instantaneous description (ID) of a PDA is represented by a triplet (q, w, s) where

- **q** is the state
- **w** is unconsumed input
- **s** is the stack contents

Turnstile Notation

The "turnstile" notation is used for connecting pairs of ID's that represent one or many moves of a PDA. The process of transition is denoted by the turnstile symbol "⊢".

Consider a PDA (Q, ∑, S, δ, q_0, I, F). A transition can be mathematically represented by the following turnstile notation −

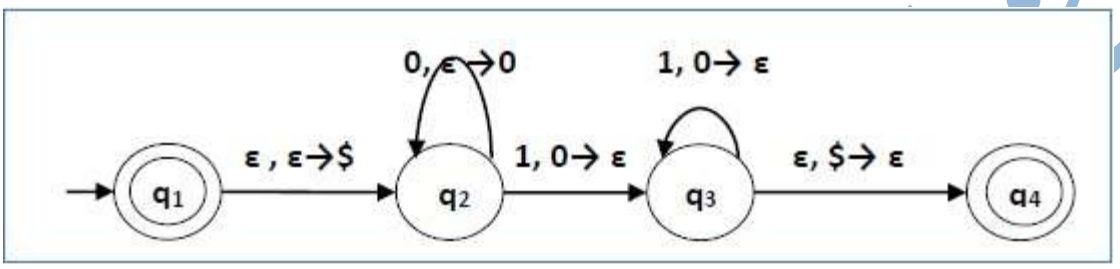
$(p, aw, T\beta) \vdash (q, w, \alpha b)$

This implies that while taking a transition from state **p** to state **q**, the input symbol **'a'** is consumed, and the top of the stack **'T'** is replaced by a new string **'α'**.

- Example

Construct a PDA that accepts $L = \{0^n 1^n \mid n \geq 0\}$

Solution:



PDA for L= {0ⁿ 1ⁿ | n≥0}

This language accepts L = {ε, 01, 0011, 000111, ... }

Here, in this example, the number of 'a' and 'b' have to be same.

- Initially we put a special symbol '$' into the empty stack.
- Then at state q2, if we encounter input 0 and top is Null, we push 0 into stack. This may iterate. And if we encounter input 1 and top is 0, we pop this 0.
- Then at state q3, if we encounter input 1 and top is 0, we pop this 0. This may also iterate. And if we encounter input 1 and top is 0, we pop the top element.
- If the special symbol '$' is encountered at top of the stack, it is popped out and it finally goes to the accepting state q4.

Pumping Lemma

Theorem

Let L be a regular language. Then there exists a constant 'c' such that for every string w in L −

$|w| \geq c$

We can break w into three strings, w = xyz, such that −

- $|y| > 0$
- $|xy| \leq c$
- For all $k \geq 0$, the string xy^kz is also in L.

Applications of Pumping Lemma

Pumping Lemma is to be applied to show that certain languages are not regular. It should never be used to show a language is regular.

- If **L** is regular, it satisfies Pumping Lemma.
- If **L** does not satisfy Pumping Lemma, it is non-regular.

Method to prove that a language L is not regular

- At first, we have to assume that **L** is regular.
- So, the pumping lemma should hold for **L**.
- Use the pumping lemma to obtain a contradiction −
  - Select **w** such that $|w| \geq c$
  - Select **y** such that $|y| \geq 1$
  - Select **x** such that $|xy| \leq c$
  - Assign the remaining string to **z.**
  - Select **k** such that the resulting string is not in **L.**

Hence L is not regular.

Problem

Prove that $L = \{a^ib^i \mid i \geq 0\}$ is not regular.

Solution −

- At first, we assume that **L** is regular and n is the number of states.
- Let w = $a^n b^n$. Thus |w| = 2n ≥ n.
- By pumping lemma, let w = xyz, where |xy| ≤ n.
- Let x = a^p, y = a^q, and z = $a^r b^n$, where p + q + r = n, p ≠ 0, q ≠ 0, r ≠ 0. Thus |y| ≠ 0.
- Let k = 2. Then $xy^2z = a^p a^{2q} a^r b^n$.
- Number of as = (p + 2q + r) = (p + q + r) + q = n + q
- Hence, $xy^2z = a^{n+q} b^n$. Since q ≠ 0, xy^2z is not of the form $a^n b^n$.
- Thus, xy^2z is not in L. Hence L is not regular.

Turing Machine

Turing machine was invented in 1936 by Alan Turing. It is an accepting device which accepts Recursive Enumerable Language generated by type 0 grammar.

There are various features of the Turing machine:

- It has an external memory which remembers arbitrary long sequence of input.
- It has unlimited memory capability.
- The model has a facility by which the input at left or right on the tape can be read easily.
- The machine can produce a certain output based on its input. Sometimes it may be required that the same input has to be used to generate the output. So in this machine, the distinction between input and output has been removed. Thus a common set of alphabets can be used for the Turing machine.

Formal definition of Turing machine

A Turing machine can be defined as a collection of 7 components:

Q: the finite set of states
∑: the finite set of input symbols
T: the tape symbol
q0: the initial state
F: a set of final states
B: a blank symbol used as an end marker for input
δ: a transition or mapping function.

The mapping function shows the mapping from states of finite automata and input symbol on the tape to the next states, external symbols and the direction for moving the tape head. This is known as a triple or a program for Turing machine.

$(q0, a) \rightarrow (q1, A, R)$

That means in q0 state, if we read symbol 'a' then it will go to state q1, replaced a by X and move ahead right(R stands for right).

Solution: Initially, it is given a finite sequence of 0's and 1's on its tape, preceded and followed by an infinity of blanks. Alternately, the TM will change a 0 to an x and then a 1 to a Y, until all 0's and 1's have been matched.

In more detail, starting at the left end of the input, it enters a loop in which it changes a 0 to an x and moves to the right over whatever 0's and Y's it sees, until it comes to a 1. It changes the to a Y, and moves left, over Y's and 0's until it finds an x. At that point, it looks for a 0 immediately to the right and if it finds one, changes it to x and repeals the process, changing a matching 1 to a Y.

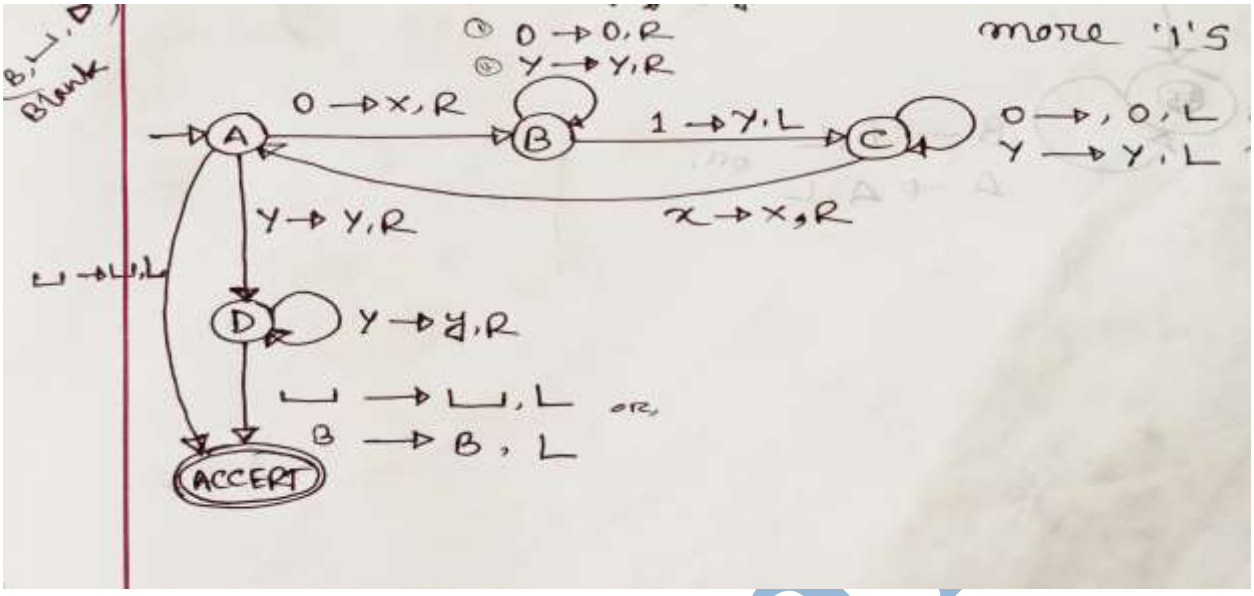
If the nonblank input is not in 0^*1^*, then the TM will eventually fail to have a next move and will die without accepting. However, if it finishes changing all the 0's to x's on the same round it changes the last 1 to a Y, then it has found its input to be of the 0^n1^n and accepts.

The formal specification of the TM is $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$

Where δ is given by the table in below:

State	0	1	X	Y	B
q_0	(q_1, X, R)	-	-	(q_3, Y, R)	-
q_1	(q_1, 0, R)	(q_2, Y, L)	-	(q_1, Y, R)	-
q_2	(q_2, 0, L)	-	(q_0, X, R)	(q_2, Y, L)	-
q_3	-	-	-	(q_3, Y, R)	(q_4, B, R)
q_4	-	-	-	-	-

Figure: A Turing machine to accept $\{0^n1^n \mid n \geq 1\}$



Chomsky Normal Form (CNF)

A CFG is in Chomsky Normal Form if the Productions are in the following forms −

- A → a
- A → BC
- S → ε

Where A, B, and C are non-terminals and **a** is terminal.

Algorithm to Convert into Chomsky Normal Form −

Step 1 − If the start symbol **S** occurs on some right side, create a new start symbol **S'** and a new production **S'→ S**.

Step 2 − Remove Null productions. (Using the Null production removal algorithm discussed earlier)

Step 3 − Remove unit productions. (Using the Unit production removal algorithm discussed earlier)

Step 4 − Replace each production $\mathbf{A \rightarrow B_1 \ldots B_n}$ where $\mathbf{n > 2}$ with $\mathbf{A \rightarrow B_1 C}$ where $\mathbf{C \rightarrow B_2 \ldots B_n}$. Repeat this step for all productions having two or more symbols in the right side.

Step 5 − If the right side of any production is in the form **A → aB** where a is a terminal and **A, B** are non-terminal, then the production is replaced by **A → XB** and **X → a**. Repeat this step for every production which is in the form **A → aB**.

Problem

Convert the following CFG into CNF

S → ASA | aB, A → B | S, B → b | ε

Solution

(1) Since **S** appears in R.H.S, we add a new state **S_0** and **S_0→S** is added to the production set and it becomes −

S_0→S, S→ ASA | aB, A → B | S, B → b | ∈

(2) Now we will remove the null productions −

B → ∈ and A → ∈

After removing B → ε, the production set becomes −

S_0→S, S→ ASA | aB | a, A → B | S | ∈, B → b

After removing A → ∈, the production set becomes −

S_0→S, S→ ASA | aB | a | AS | SA | S, A → B | S, B → b

(3) Now we will remove the unit productions.

After removing S → S, the production set becomes −

S_0→S, S→ ASA | aB | a | AS | SA, A → B | S, B → b

After removing S_0→ S, the production set becomes −

S_0→ ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA

A → B | S, B → b

After removing A→ B, the production set becomes −

S_0 → ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA

A → S | b

B → b

After removing A→ S, the production set becomes −

S_0 → ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA

A → b |ASA | aB | a | AS | SA, B → b

(4) Now we will find out more than two variables in the R.H.S

Here, $S_0 \rightarrow$ ASA, S $\rightarrow$ ASA, A $\rightarrow$ ASA violates two Non-terminals in R.H.S.

Hence we will apply step 4 and step 5 to get the following final production set which is in CNF −

$S_0 \rightarrow$ AX | aB | a | AS | SA

S $\rightarrow$ AX | aB | a | AS | SA

A $\rightarrow$ b |AX | aB | a | AS | SA

B $\rightarrow$ b

X $\rightarrow$ SA

(5) We have to change the productions $S_0 \rightarrow$ aB, S $\rightarrow$ aB, A $\rightarrow$ aB

And the final production set becomes −

$S_0 \rightarrow$ AX | YB | a | AS | SA

S $\rightarrow$ AX | YB | a | AS | SA

A $\rightarrow$ b A $\rightarrow$ b |AX | YB | a | AS | SA

B $\rightarrow$ b

X $\rightarrow$ SA

Y $\rightarrow$ a

*****If there is any oversight correct it by your own.**