In computer science, a pushdown automaton (PDA) is a type of automaton that employs a stack. Pushdown automata are used in theories about what can be computed by machines. They are more capable than finite-state machines but less capable than Turing machines. Deterministic pushdown automata can recognize all deterministic context-free languages while nondeterministic ones can recognize all context-free languages. Mainly the former are used in parser design.

The term "pushdown" refers to the fact that the stack can be regarded as being "pushed down" like a tray dispenser at a cafeteria, since the operations never work on elements other than the top element. A stack automaton, by contrast, does allow access to and operations on deeper elements. Stack automata can recognize a strictly larger set of languages than pushdown automata. A nested stack automaton allows full access, and also allows stacked values to be entire sub-stacks rather than just single finite symbols.

Here, it describes the nondeterministic pushdown automaton.

Basic Structure of 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 –

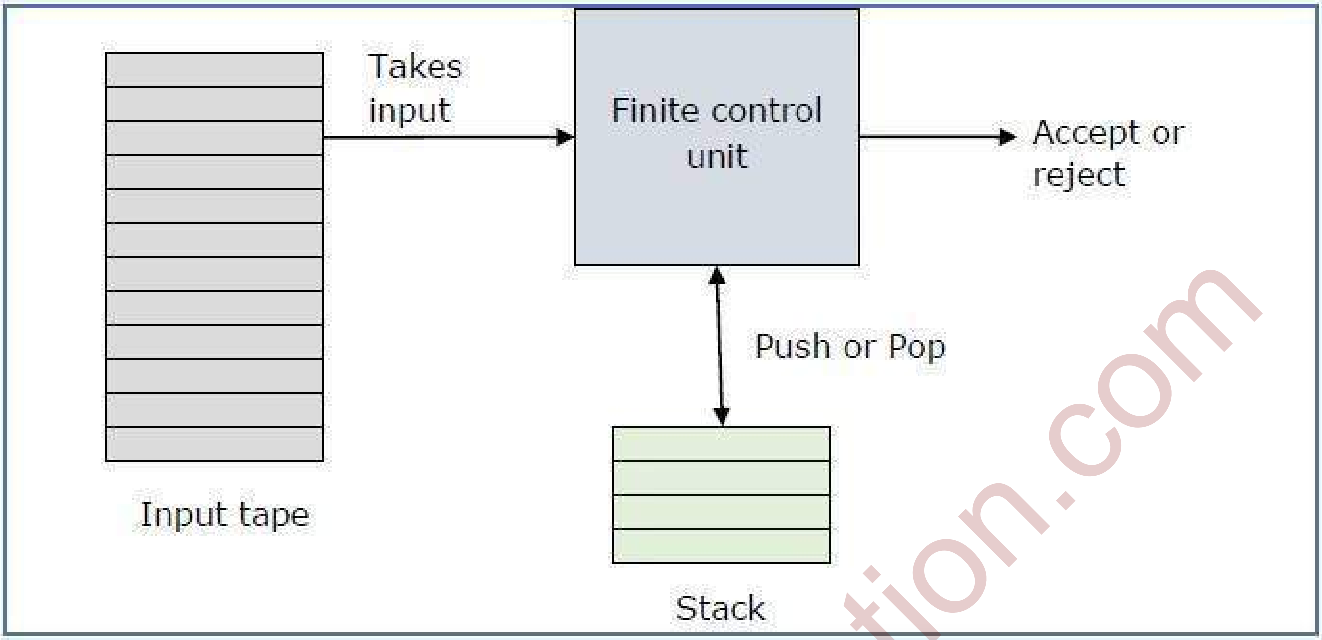
- an input tape,
- a control unit, and
- a stack with infinite size.

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.

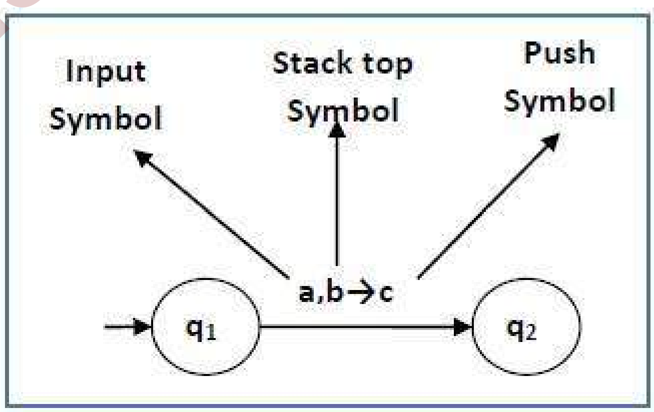
A PDA may or may not read an input symbol, but it has to read the top of the stack in every transition.



A PDA can be formally described as a 7-tuple (Q, ∑, S, δ, 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’.

Note − If we want zero or more moves of a PDA, we have to use the symbol ($\vdash^*$) for it.

Final State Acceptability

In final state acceptability, a PDA accepts a string when, after reading the entire string, the PDA is in a final state. From the starting state, we can make moves that end up in a final state with any stack values. The stack values are irrelevant as long as we end up in a final state.

For a PDA (Q, ∑, S, δ, q_0, I, F), the language accepted by the set of final states F is −

$L(PDA) = \{w \mid (q_0, w, I) \vdash^* (q, \varepsilon, x), q \in F\}$

For any input stack string x.

Empty Stack Acceptability

Here a PDA accepts a string when, after reading the entire string, the PDA has emptied its stack.

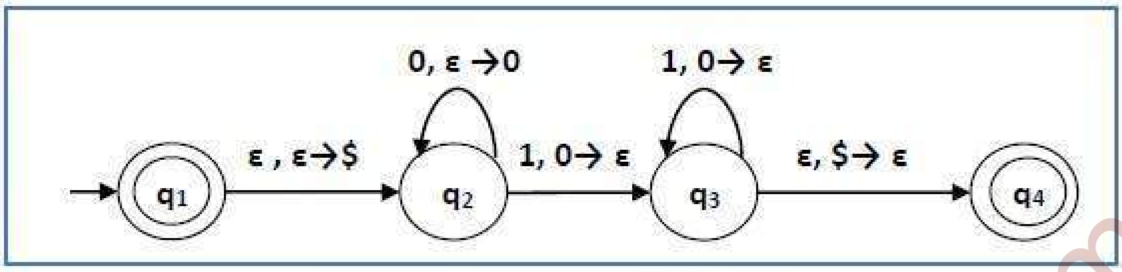
For a PDA (Q, ∑, S, δ, q_0, I, F), the language accepted by the empty stack is−

$L(PDA) = \{w \mid (q_0, w, I) \vdash^* (q, \varepsilon, \varepsilon), q \in Q\}$

Example

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

Solution



PDA for L= $\{0^n 1^n \mid n \geq 0\}$

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

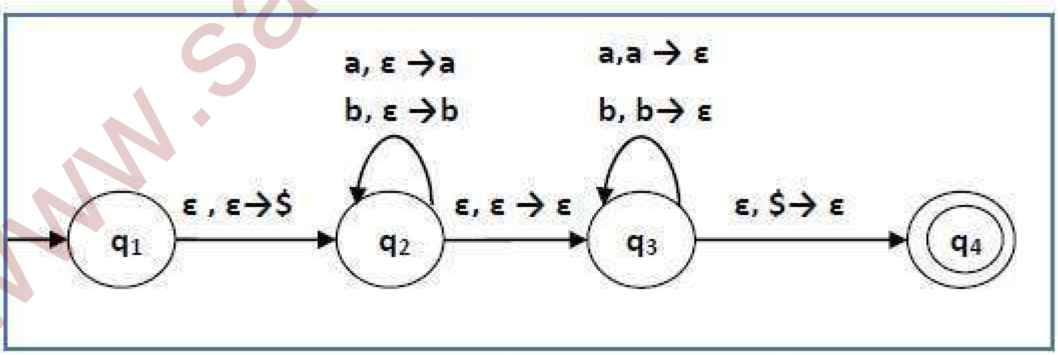
Here, in this example, the number of 'a' and 'b' have to be same.

- Initially we put a special symbol '$' into the empty stack.
- Then at state q_2, if we encounter input 0 and top is Null, we push 0 into stack. This may iterate. And if we encounter input 1 and top is 0, we pop this 0.
- Then at state q_3, if we encounter input 1 and top is 0, we pop this 0. This may also iterate. And if we encounter input 1 and top is 0, we pop the top element.
- If the special symbol '$' is encountered at top of the stack, it is popped out and it finally goes to the accepting state q_4.

Example

Construct a PDA that accepts L= $\{ww^R \mid w = (a+b)^* \}$

Solution



PDA for L= $\{ww^R \mid w = (a+b)^*\}$

Initially we put a special symbol '$' into the empty stack. At state q_2, the w is being read. In state q_3, each 0 or 1 is popped when it matches the input. If any other input is given, the PDA will go to a dead state. When we reach that special symbol '$', we go to the accepting state q_4.

Equivalence of CFL and PDA:

If a grammar **G** is context-free, we can build an equivalent nondeterministic PDA which accepts the language that is produced by the context-free grammar**G**. A parser can be built for the grammar **G**. Also, if P is a pushdown automaton, an equivalent context-free grammar G can be constructed where

L(G) = L(P)

Algorithm to find PDA corresponding to a given CFG

Input − A CFG, G= (V, T, P, S)

Output − Equivalent PDA, P= (Q, ∑, S, δ, q0, I, F)

Step 1 Convert the productions of the CFG into GNF.

Step 2 The PDA will have only one state {q}.

Step 3 The start symbol of CFG will be the start symbol in the PDA.

Step 4 All non-terminals of the CFG will be the stack symbols of the PDA and all the terminals of the CFG will be the input symbols of the PDA.

Step 5 For each production in the form $A \rightarrow aX$ where a is terminal and A, X are combination of terminal and non-terminals, make a transition δ (q, a, A).