RGPVNOTES.IN

Program : **B.Tech**

Subject Name: **Theory of Computation**

Subject Code: **IT-503**

Semester: **5th**

Department of Information Technology
Subject Notes
IT503 (A) - Theory of Computation
B.Tech, IT-5th Semester

Unit IV

Syllabus: Introduction of PDA, formal definition, closure property of PDA, examples of PDA, Deterministic Pushdown Automata, NPDA, conversion PDA to CFG, conversion CFG to PDA.

Unit Objective: Designing problems on Pushdown Automata and conversion of grammar to PDA, PDA to Grammar.

Unit-IV: Pushdown Automata

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.

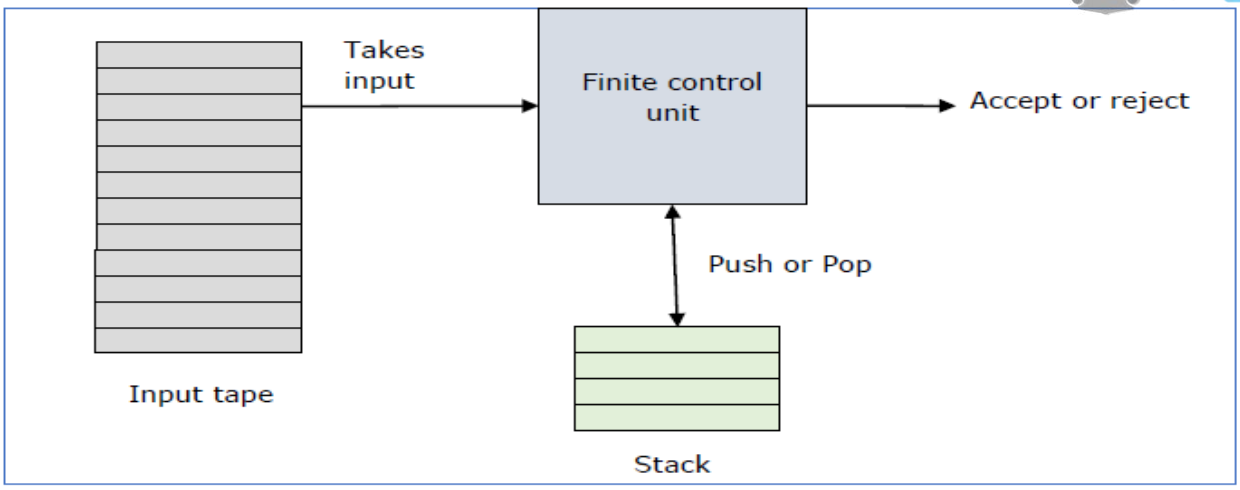

Figure 4.1: Pushdown Automata

Applications of PDA:

1. Online Transaction process system.
2. Used in compiler design(parser design for syntactic analysis)
3. Tower of Hanoi (Recursive Solution)

Formal Definition of PDA:

A deterministic pushdown automaton is a 7 -tuples M = (Q, Σ, Γ, δ, q0, Z0, F), where

- Q is a finite set of states,
- Σ is a finite set of tape alphabet or input symbol
- Γ is a finite set of stack alphabet
- q0 is initial state , q0 is an element of Q
- Z0 is Initial symbol on top of stack
- F set of final state which is sub set of Q.
- δ is transition function which maps (Q x Σ x Γ) into Q x Γ*

Transition of PDA:

Transition function of PDA is denoted as δ(q, a, X) = (p, Y) which specified transition in PDA is function of three components :

1. Present state of PDA, q
2. Symbol of input alphabet, a, being read by PDA.

3. Symbol at top of stack, X.

In every transition

- PDA enters into new state p or remain into same state p = q
- if a = ε than no symbol is consumed
- If Y = zX, symbol z is pushed into the stack at the top.
- If Y = ε, Symbol X at the top of stack is popped.
- If Y = X, No change of symbol at top of stack.

Example: The following diagram shows a transition in a PDA from a state q_1 to state q_2, labeled as "a,b/c"

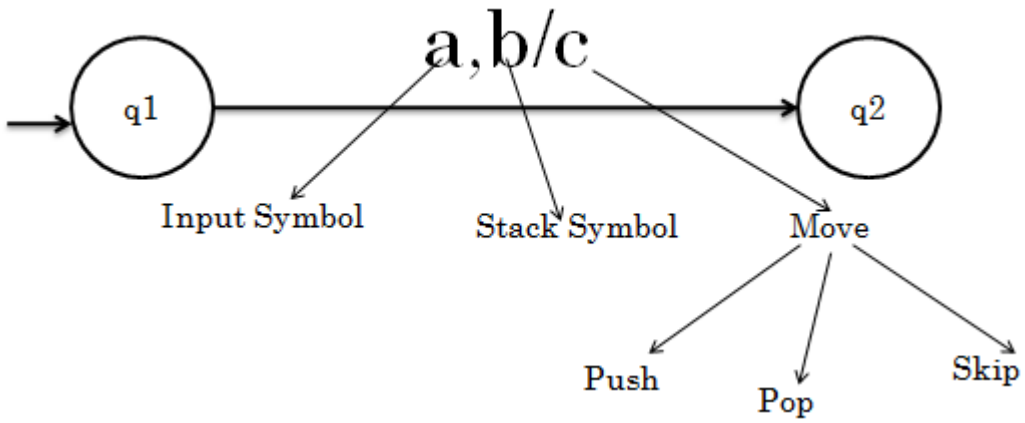

Figure 4.2: Example of PDA

Representation of PDA:

Corresponding to transition function δ(q, a, X) = (p,Y) transition diagram will have

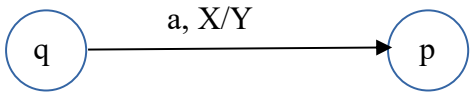

Figure 4.3: Representation of PDA

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, Σ, S, δ, q0, 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 (⊢*) for it.

Deterministic PDA:

A PDA is said to be deterministic if and only if following conditions are met

1. δ(q, a, X) has at most one transition.
2. δ(q, ε , X) = Φ

Acceptance of PDA:

There are two different ways to define PDA acceptability.

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, Σ, S, δ, q0, I, F), the language accepted by the set of final states F is:

$$L(PDA) = \{w \mid (q0, w, I) \vdash^* (q, \varepsilon, x), q \in F\}$$

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, Σ, S, δ, q0, I, F), the language accepted by the empty stack is:

$$L(PDA) = \{w \mid (q0, w, I) \vdash^* (q, \varepsilon, \varepsilon), q \in Q\}$$

Example: Construct a PDA that accepts L= $\{0^n1^n \mid n \geq 0\}$

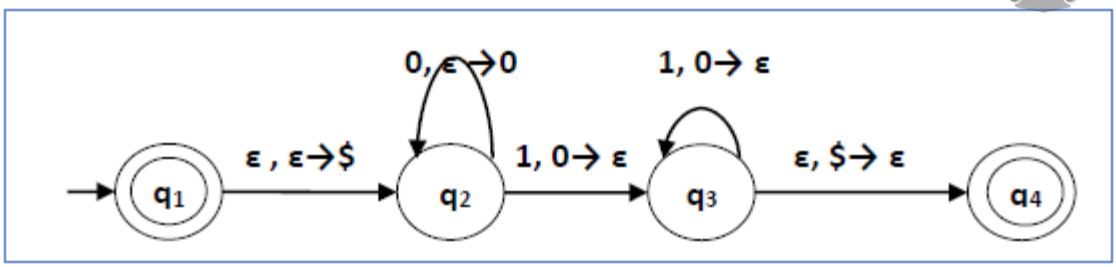

Figure 4.4: Example of PDA

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.

Non-deterministic PDA:

A non-deterministic pushdown automaton (NPDA), or just pushdown automaton (PDA) is a variation on the idea of a non-deterministic finite automaton (NDFA). Unlike an NDFA, a PDA is associated with a stack (hence the name pushdown). The transition function must also take into account the "state" of the stack.

Formally defined, a pushdown automaton M is a 7-tuple M=(Q,Σ,Γ,T,q0,⊥,F), where Q,Σ,q0, and F, like those in an NDFA, are the set of states, the input alphabet, the start state, and the set of final states respectively. Γ is the stack alphabet, specifying the set of symbols that can be pushed onto the stack. Γ is not necessarily disjoint from Σ. ⊥ is an element of Γ called the start stack symbol. The transition function is: $T:Q\times(\Sigma\cup\{\lambda\})\times\Gamma\rightarrow\mathcal{P}(Q\times\Gamma^*)$.

PDA & Context Free Grammar:

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, Σ, 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→ aX where a is terminal and A, X are combination of terminal and non-terminals, make a transition δ (q, a, A).

Example: Construct a PDA from the following CFG.

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

where the productions are:

$$S \rightarrow XS \mid €, A \rightarrow aXb \mid Ab \mid ab$$

Solution:

Let the equivalent PDA,

P = ({q}, {a, b}, {a, b, X, S}, δ, q, S)

where δ:

δ (q, € , S) = {(q, XS), (q, €)}

δ (q, € , X) = {(q, aXb), (q, Xb), (q, ab)}

δ (q, a, a) = {(q, €)}

δ (q, 1, 1) = {(q, €)}

Algorithm to find CFG corresponding to a given PDA:

Input: A CFG, G= (V, T, P, S)

Output: Equivalent PDA, P = (Q, Σ, S, δ, q0, I, F) such that the non- terminals of the grammar G will be $\{X_{wx} \mid w,x \in Q\}$ and the start state will be Aq0,F.

Step 1: For every w, x, y, z ∈ Q, m ∈ S and a, b ∈ Σ, if δ (w, a, €) contains (y, m) and (z, b, m) contains (x, €), add the production rule $X_{wx} \rightarrow a X_{yz} b$ in grammar G.

Step 2: For every w, x, y, z ∈ Q, add the production rule $X_{wx} \rightarrow X_{wy} X_{yx}$ in grammar G.

Step 3: For w ∈ Q, add the production rule Xww→ € in grammar G.

RGPVNOTES.IN

We hope you find these notes useful.

You can get previous year question papers at https://qp.rgpvnotes.in .

If you have any queries or you want to submit your study notes please write us at rgpvnotes.in@gmail.com

LIKE & FOLLOW US ON FACEBOOK
facebook.com/rgpvnotes.in