ABHYASMITRA.IN
STUDY NOTES

THEORY OF COMPUTATIONS

Unit IV — Push Down Automata (PDA)

Topics Covered (09 Hours)

Introduction to Stack-based Pushdown Automata
Formal 7-Tuple Definition & Mathematical Operations
Nondeterministic Pushdown Automata (NPDA) Architecture
Acceptance Modes: Final State vs. Empty Stack
Equivalence Proofs for Acceptance Mechanisms
Equivalence of PDAs and Context Free Grammars (CFGs)
Converting Context Free Grammars directly to PDAs
Pushdown Automata vs. Context Free Languages (CFLs)
Industrial Applications of PDAs (Compilers & Parsers)
Introduction to Post Machines (Queue-based Automata)

Compiled in a simple, easy-to-understand, book style format
for students of Computer Science and Engineering

TABLE OF CONTENTS

4.1 Introduction & Formal Definition of PDA

Finite automata have a major limitation: their memory is restricted to a finite number of states. Because of this, they cannot recognize languages like L = {a^n b^n | n ≥ 0} which require counting. A Pushdown Automaton (PDA) solves this limitation by adding an auxiliary memory structure in the form of a Stack. The stack provides an infinite storage capacity, but it is accessed in a Last-In, First-Out (LIFO) manner.

Pushdown Automaton Formal Definition (7-Tuple)

A Pushdown Automaton is mathematically defined as a 7-tuple:

M = (Q, Σ, Γ, δ, q0, Z0, F)

Where:

1. Q is a finite, non-empty set of internal States.

2. Σ is a finite set of input Symbols (Input Alphabet).

3. Γ is a finite set of Stack Symbols (Stack Alphabet).

4. δ is the Transition Function mapping Q x (Σ ∪ {ε}) x Γ → P(Q x Γ).*

5. q0 ∈ Q is the initial Start State.

6. Z0 ∈ Γ is the initial Stack symbol (bottom of stack marker).

7. F ⊆ Q is the set of Final (Accepting) States.

The Transition Function δ

The transition function defines the behavior of the PDA. When the machine is in state q, reading input symbol 'a' (or ε) and having stack symbol 'X' at the top of the stack, it transitions to a new state p and replaces 'X' with a string of stack symbols γ. If γ is ε, it is a POP operation. If γ is a double symbol (like YX), it is a PUSH operation.

4.2 Acceptance Modes & Equivalence

A PDA can be designed to accept input strings in one of two equivalent ways:

- **1. Acceptance by Final State:** The PDA accepts a string if, after reading the entire input, the machine reaches a final state ($q \in F$). The contents of the stack do not matter.
- **2. Acceptance by Empty Stack:** The PDA accepts a string if, after reading the entire input, the machine has popped all symbols from the stack, including the initial marker Z0 (the stack becomes empty). The final state of the machine does not matter.

Equivalence Proof Outline

Acceptance by Final State and Empty Stack are computationally equivalent. We can convert between them using specific algorithms:

- **Final State to Empty Stack:** Create a new start state q0' that pushes a special bottom marker X0 onto the stack, then transitions to the old start state q0. Create a new dummy state pe. For every final state in the old PDA, add an ε-transition to pe. From state pe, add loops that pop all remaining symbols from the stack until only X0 is left, pop X0, and terminate.
- **Empty Stack to Final State:** Create a new start state q0' that pushes X0 onto the stack, then transitions to q0. Create a new final state qf. If the machine ever pops X0 (meaning the old stack would be empty), transition to qf immediately.

4.3 Equivalence of PDA and CFG

A language is context-free if and only if it is accepted by a Pushdown Automaton. We can convert any Context Free Grammar into an equivalent PDA that parses strings using the stack.

Converting CFG to PDA (Empty Stack Mode)

To construct a PDA from a CFG $G = (V, \Sigma, R, S)$, we create a machine with a single state q. The stack alphabet is $\Gamma = V \cup \Sigma$, and the initial stack symbol is S. The transitions are built using two rules:

- **Rule 1 (Variable Expansion):** For every variable $A \in V$ and production rule $A \rightarrow \alpha$, add the transition $\delta(q, \varepsilon, A) = \{(q, \alpha)\}$. This simulates leftmost derivation steps on the stack.
- **Rule 2 (Terminal Match):** For every terminal symbol 'a' $\in \Sigma$, add the transition $\delta(q, a, a) = \{(q, \varepsilon)\}$. This pops matching terminals from the stack as they are read from the input.

Introduction to Post Machines

A Post Machine is an alternative abstract machine model. Instead of a LIFO stack, a Post Machine uses a FIFO (First-In, First-Out) Queue for memory. A Post Machine reads symbols from the front of the queue and appends new symbols to the back of the queue. Post machines are computationally equivalent to Turing Machines, making them much more powerful than standard PDAs.

Unit Summary

This unit explored Pushdown Automata and their equivalence to CFGs.

- A Pushdown Automaton adds a LIFO stack memory to a finite state machine.
- Formal PDAs are defined using a 7-tuple. Transitions pop or push stack symbols.
- PDAs can accept strings by reaching a Final State or by emptying their Stack.
- Context Free Grammars can be compiled directly into equivalent empty-stack PDAs.
- Post Machines use a FIFO queue instead of a stack, making them equivalent in power to Turing Machines.