BITS Pilani
Pilani Campus

# Pushdown Automaton and its Equivalence with CFG

Shashank Gupta
Assistant Professor
Department of Computer Science and Information Systems

# Formal Definition of Pushdown Automata (PDA)

A PDA consist of 6 tuples (Q, Σ, Γ, δ, q0, F) where

- Q is finite set of states.
- Σ is the input alphabet.
- Γ is the stack alphabet.
- δ: Transition relation, which is a finite subset of $(Q \times (\Sigma \cup \{\in\}) \times \Gamma_{\in}) \times Q \times \Gamma^*$
- q0: Start State
- F is the set of accept states.

# Transition function of PDA

Input (State, Input alphabet, Stack alphabet)

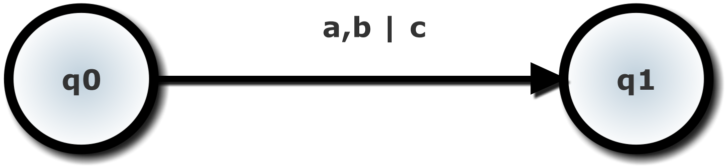
- State p
- $a_i \in \Sigma_\epsilon$
- $X \in \Gamma^*$

Output are the pairs of form (State, Stack Alphabet)

# Pushdown Automata

Due to non-determinism of the PDA, there can be multiple transitions on the same tuple (p,a,X)

Transitions can be denoted as follows:



# Pushdown Automata

A PDA $P = (Q, \Sigma, \Gamma, \delta, q0, F)$ is said to accept a string $w \in \Sigma^*$ if there exists

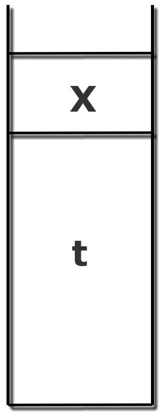
- A sequence of symbols $a_1, a_2$, -----, $a_m \in \Sigma_\epsilon$
- States $r_0, r_1$, ----- $r_m \in Q$
- Strings $s_0, s_1$,-----$s_m \in \Gamma^*$ s.t.

Initial Condition: $w = a_1, a_2$, -----, $a_m$

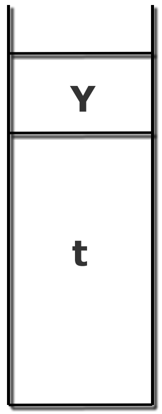
$r_0 = q_0$ and $s_0 = z_0$

- $\forall i$, if $\delta(r_{i-1}, a_i, X) \in (r_i, Y)$ then $s_{i-1} = Xt$ and $s_i = Yt$ for some $t \in \Gamma^*$ and $X, Y \in \Gamma_\epsilon$
- $r_m \in F$

# Pushdown Automata




Before the $i^{th}$ Step




After the $i^{th}$ Step

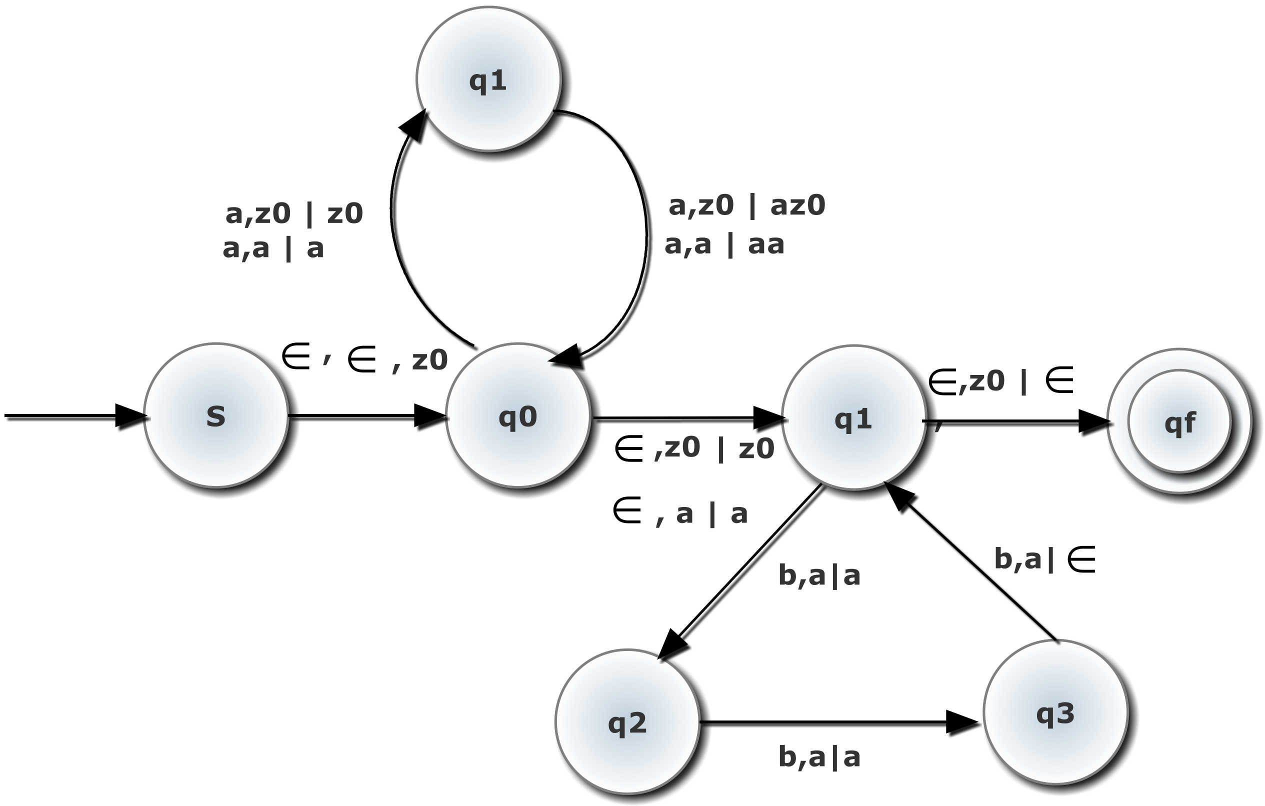
# Example

Design a PDA for the following language

$$L = a^{2p}\, b^{3p} \mid p \geq 0 \}$$

# Example (Continued…..)

Design a PDA for the following language $L = a^{2p}\, b^{3p} \mid p \geq 0 \}$



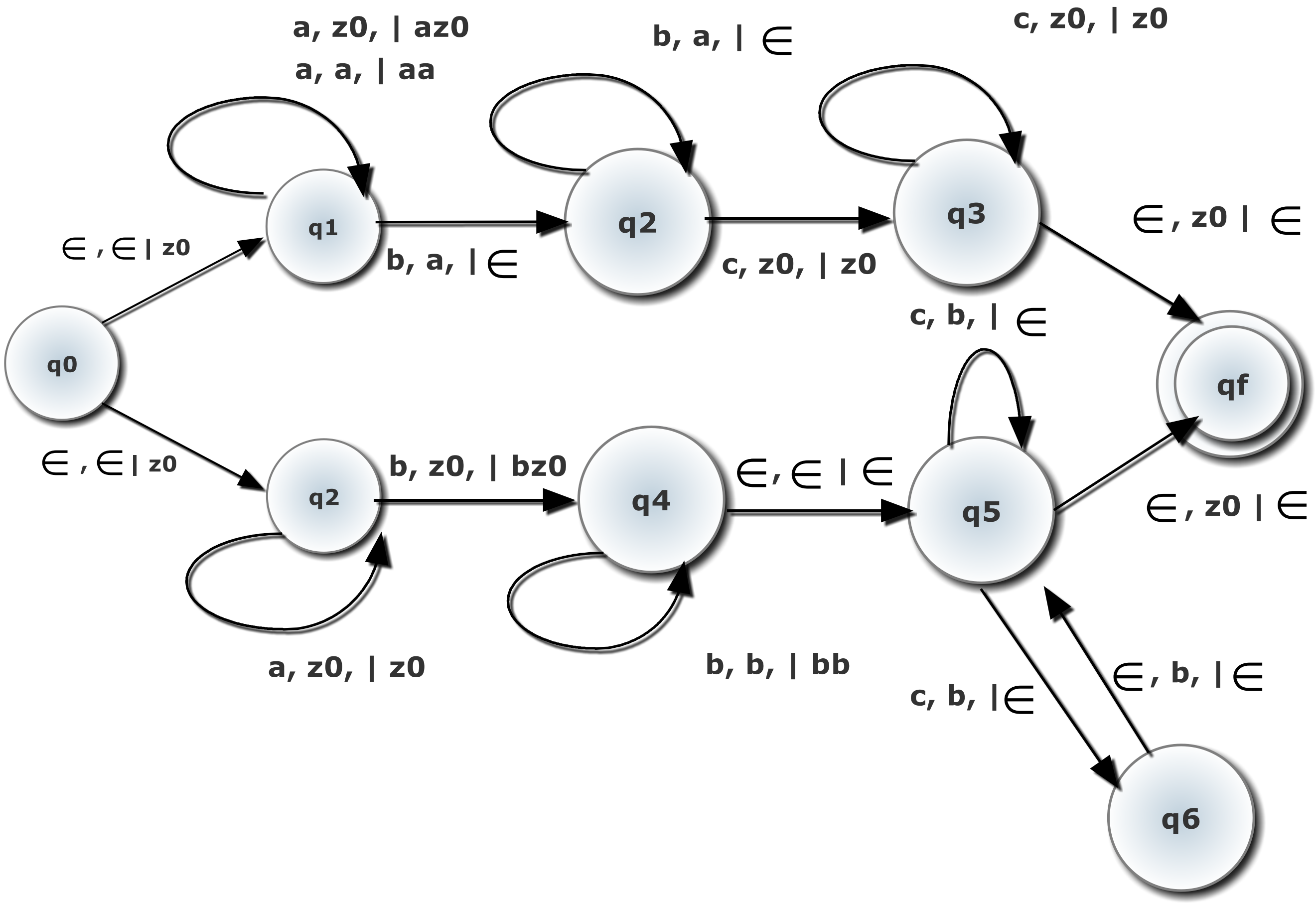
# More Designing Examples

Consider a PDA for the following language

$$L = \{a^i\, b^j\, c^k \mid i = j \text{ or } k \leq j \leq 2k\}$$

# Example (Continued…..)

$L = \{a^i b^j c^k \mid i = j \text{ or } k \leq j \leq 2k\}$



# Equivalence between CFG and PDA

# CFG to PDA

Each Context-free language is accepted by PDA

Let G = (V,T,P,S) be a CFG, we must construct a PDA M s.t.

- L(M) = L(G)

# CFG to PDA

Given a CFG G, we construct a PDA that simulates the leftmost derivations of G.

Any left-sentential form that is not a terminal string can be written as $xA\alpha$,

- where A is the leftmost variable, x is whatever terminals appear to its left, and $\alpha$ is the string of terminals and variables that appear to the right of A

# CFG to PDA

The idea behind the construction of a PDA from a grammar is to have the PDA simulate the sequence of left-sentential forms that the grammar uses to generate a given terminal string w

PDA has

- 2 states p and q.
- **I/P Symbols**: terminals of G
- **Stack Symbols**: all symbols of G.
- **Start Symbol**: start symbol of G.

# CFG to PDA

Given input w, PDA will step through a leftmost derivation of w from the start symbol S.

Since PDA can’t know what this derivation is, or even what the end of w is, it uses non-determinism to “guess” the production to use at each step

# CFG to PDA

At each step, PDA P represents some left sentential form (step of a leftmost derivation).

If the stack of P is $\alpha$, and P has so far consumed x from its input, then P represents left-sentential form $x\alpha$.

- At empty stack, the input consumed is a string in L(G).

# Conversion from CFG to PDA

Let $G = (N, T, P, S)$ be a CFG.

We construct a PDA $P = (Q, \Sigma, \Gamma, \delta, q0, F)$:

$Q = \{p,q\}$

$q0 = p$

$\Sigma = T$ -Input Alphabet is set of terminals I

$\Gamma = N \cup T$ Stack alphabet is terminals and non-terminals

$F = q$

# Definition of Transition Function of PDA

Initially, $\delta(p, \in, \in) = (q, S)$

For each variable A, $\delta(q, \in, A) = \{(q, \beta) | A \rightarrow \beta$ is a production of P$\}$.

For each terminal a, $\delta(q, a, a) = \{(q, \in)\}$.

- At empty stack and on final state q, the input consumed is a string in L(G).

# Example

Consider the grammar G = (V, T, P, S) with V = {S}, T = {a, b, c}, and P = {S → aSa, S → bSb, S → c}, which generates the language $\{wcw^R | w \in \{a, b\}*\}$. Design the corresponding pushdown automaton (acceptance by empty stack)

a) $\delta(p, \in, \in) = \{(q, S)\}$

b) $\delta(q, \in, S) = \{(q, aSa),(q, bSb),(q, c)\}$

c) $\delta(q, a, a) = (q, \in)$

d) $\delta(q, b, b) = (q, \in)$

e) $\delta(q, c, c) = (q, \in)$

# Acceptance of String by Empty Stack and Final State q

innovate achieve lead

$\delta(p, abcba, \in)$
$\delta(q, abcba, S)$
$\delta(q, abcba, aSa)$
$\delta(q, bcba, Sa)$
$\delta(q, bcba, bSba)$
$\delta(q, cba, Sba)$
$\delta(q, cba, cba)$
$\delta(q, ba, ba)$
$\delta(q, a, a)$
$\delta(q, \in, \in)$

a) $\delta(p, \in, \in) = \{(q, S)\}$
b) $\delta(q, \in, S) = \{(q, aSa),(q, bSb),(q, c)\}$
c) $\delta(q, a, a) = (q, \in)$
d) $\delta(q, b, b) = (q, \in)$
e) $\delta(q, c, c) = (q, \in)$