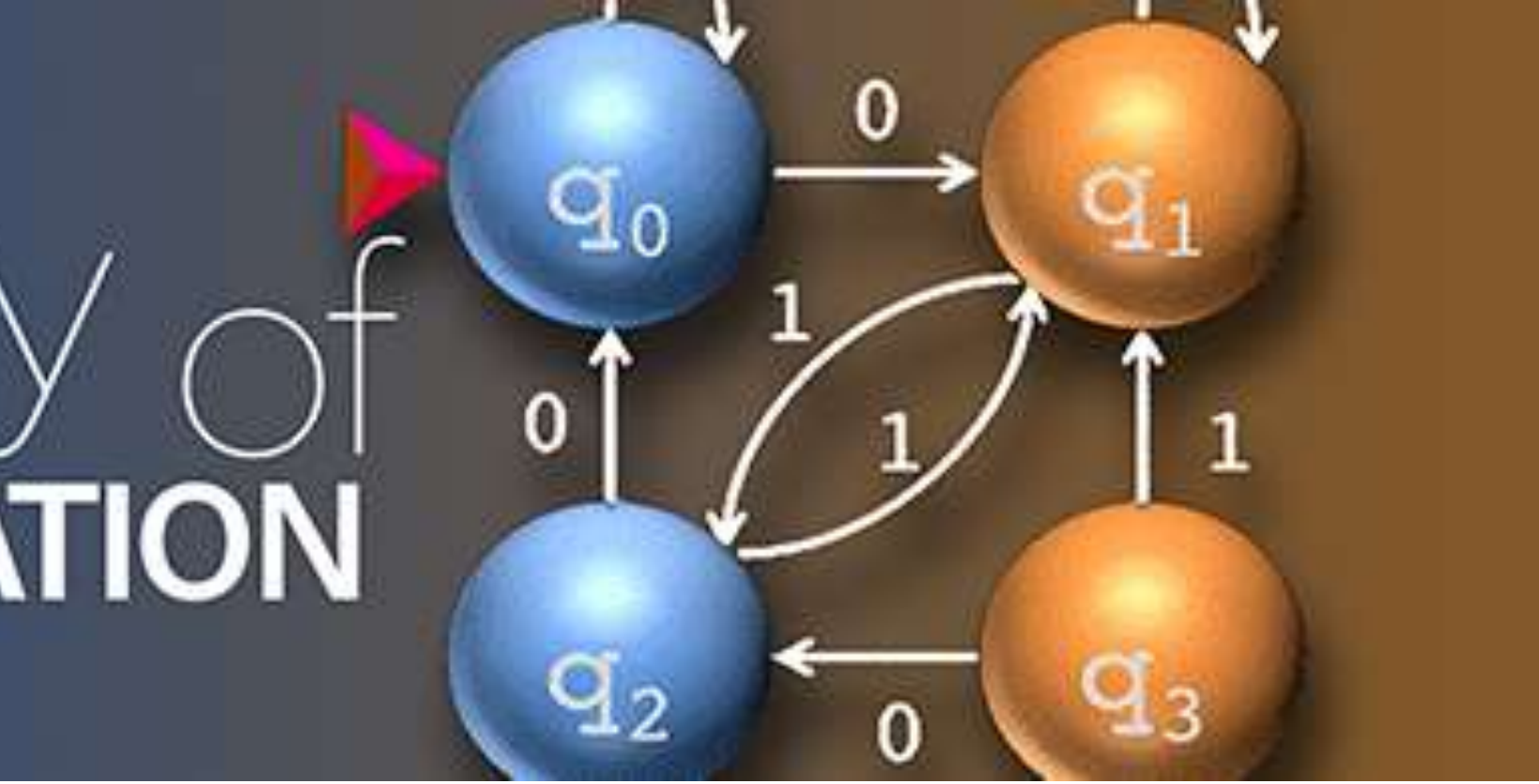
CSE 305

Theory of COMPUTATION



Lecture 26

Pushdown Automata (1)

Md. Mijanur Rahman, Prof. Dr.

Dept. of Computer Science and Engineering, Jatiya Kabi Kazi Nazrul Islam University, Bangladesh.

www.mijanrahman.com

Contents

Pushdown Automata

- Pushdown Automata (PDA)
- Components of PDA
- Formal Definition of PDA
- Instantaneous Description and Notation
- Examples
- PDA Acceptance
- Non-deterministic PDA
- PDA & CFG

Pushdown Automata (PDA)

- Pushdown automata is a way to implement a CFG in the same 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.
- Pushdown automata is simply an NFA augmented with an "external stack memory". The addition of stack is used to provide a last-in-first-out memory management capability to Pushdown automata.
- Pushdown automata can store an unbounded amount of information on the stack. It can access a limited amount of information on the stack.
- A PDA can push an element onto the top of the stack and pop off an element from the top of the stack. To read an element into the stack, the top elements must be popped off and are lost.

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 PDA is more powerful than FA. Any language which can be acceptable by FA can also be acceptable by PDA. PDA also accepts a class of language which even cannot be accepted by FA. Thus PDA is much more superior to FA.

- It is a finite automata with stack which helps Pushdown automata to recognize Context Free Languages. A DFA can remember a finite amount of information, but a PDA can remember an infinite amount of information.

Pushdown Automata (PDA)

- Basically a pushdown automaton is −

  "Finite state machine" + "a stack“

- 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.

Components of PDA

PDA Components:

- **Input tape:** The input tape is divided in many cells or symbols. The input head is read-only and may only move from left to right, one symbol at a time.
- **Finite control:** The finite control has some pointer which points the current symbol which is to be read.
- **Stack:** The stack is a structure in which we can push and remove the items from one end only. It has an infinite size. In PDA, the stack is used to store the items temporarily.

- The following figure shows schematic diagram of PDA.

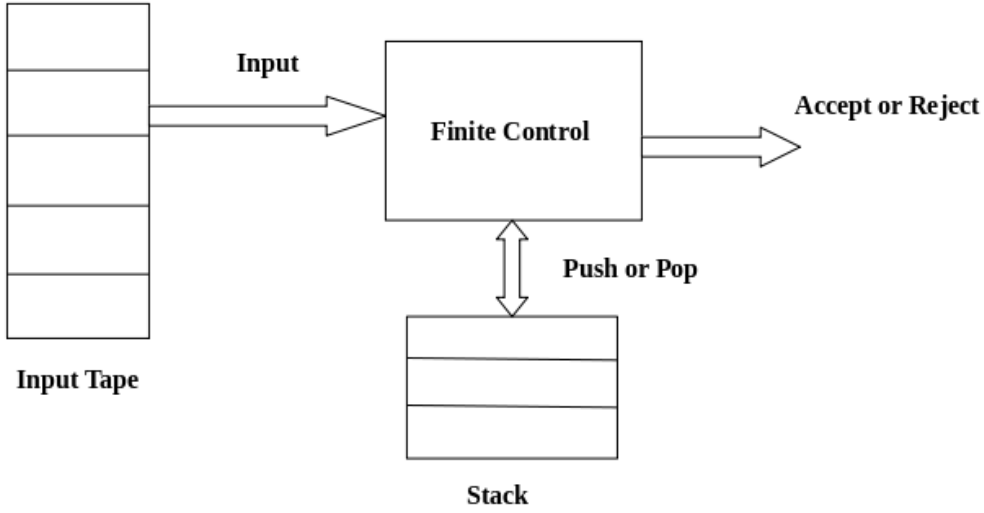


Fig: Pushdown Automata

Formal Definition of PDA

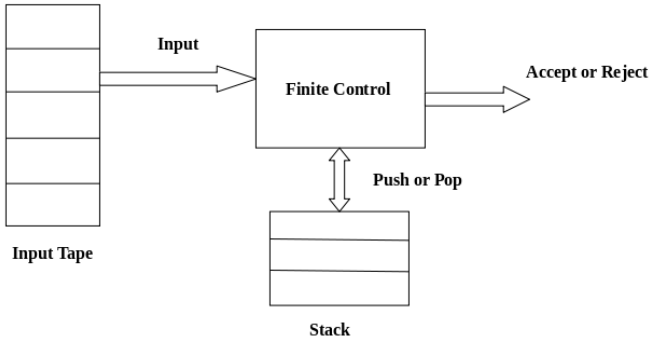


Fig: Pushdown Automata

Formal definition of PDA:

- The PDA can be defined as a collection of 7 components, such as a PDA is defined by M = {**Q, ∑, Γ, δ, q_0, Z, F**}, where:

  Q: the finite set of states

  ∑: the input set

  Γ: a stack symbol which can be pushed and popped from the stack

  δ: mapping function which is used for moving from current state to next state.

  q0: the initial state

  Z: a start symbol which is in Γ.

  F: a set of final states

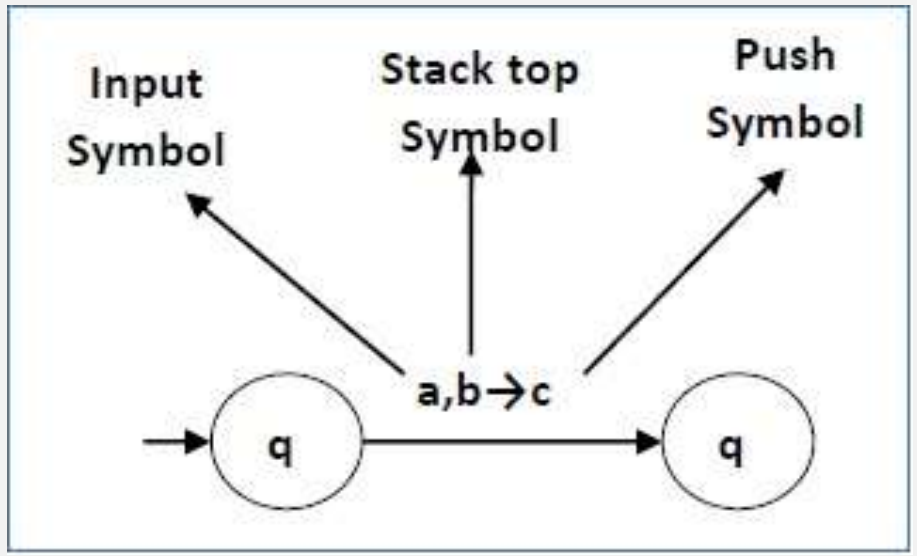
Transition in a PDA

Transition Function of PDA:

- **δ:** mapping function which is used for moving from current state to next state.
- It maps $Q \times \{\Sigma \cup \in\} \times \Gamma$ into $Q \times \Gamma^*$.
- In a given state, PDA will read input symbol and stack symbol (top of the stack) and move to a new state and change the symbol of stack.

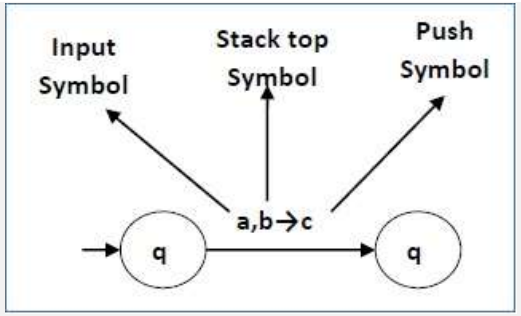
Transition in a PDA

- The following diagram shows a transition in a PDA from a state $\mathbf{q_1}$ to state $\mathbf{q_2}$, labeled as **a, b → c**.



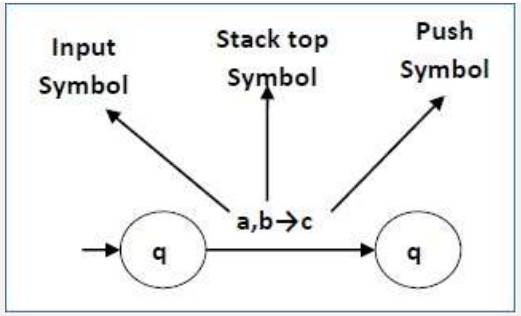
- This means at state $\mathbf{q_1}$, if we encounter an input string **'a'** and top symbol of the stack is **'b'**, then we pop **'b'**, push **'c'** on top of the stack and move to state $\mathbf{q_2}$.

Instantaneous Description



- Instantaneous Description (ID) is an informal notation of how a PDA "computes" a input string and make a decision that string is accepted or rejected.
- A ID is a triple (q, w, α), where:
  1. q is the current state.
  2. w is the remaining input.
  3. α is the stack contents, top at the left.

Turnstile Notation



- ⊢ sign is called a “turnstile notation” and represents one move.
- ⊢* sign represents a sequence of moves.

  For example, (p, b, T) ⊢ (q, w, α)
- This implies that while taking a transition from state p to state q, the input symbol ‘b’ is consumed, and the top of the stack ‘T’ is replaced by a new string ‘α’

Example: Pushdown Automata(PDA)

Example-1 : Design a PDA for accepting a language $\{a^n b^{2n} \mid n>=1\}$.

- **Solution:** In this language, n number of a's should be followed by 2n number of b's. Hence, we will apply a very simple logic, and that is if we read single 'a', we will push two a's onto the stack. As soon as we read 'b' then for every single 'b' only one 'a' should get popped from the stack.
- The ID can be constructed as follows:

  $\delta(q0, a, Z) = (q0, aaZ)$

  $\delta(q0, a, a) = (q0, aaa)$
- Now when we read b, we will change the state from q0 to q1 and start popping corresponding 'a'. Hence,

  $\delta(q0, b, a) = (q1, \varepsilon)$
- Thus this process of popping 'b' will be repeated unless all the symbols are read. Note that popping action occurs in state q1 only.

  $\delta(q1, b, a) = (q1, \varepsilon)$
- After reading all b's, all the corresponding a's should get popped. Hence when we read ε as input symbol then there should be nothing in the stack. Hence the move will be:

  $\delta(q1, \varepsilon, Z) = (q2, \varepsilon)$

Example: Pushdown Automata(PDA)

- **Thus, the PDA,** M = {**Q, ∑, Γ, δ, q_0, Z, F**}, where Q = {q0, q1, q2} and Σ = { a, b }, Γ = {a, Z}, F= {q2} and δ is given by the following ID:

  δ(q0, a, Z) = (q0, aaZ)

  δ(q0, a, a) = (q0, aaa)

  δ(q0, b, a) = (q1, ε)

  δ(q1, b, a) = (q1, ε)

  δ(q1, ε, Z) = (q2, ε)

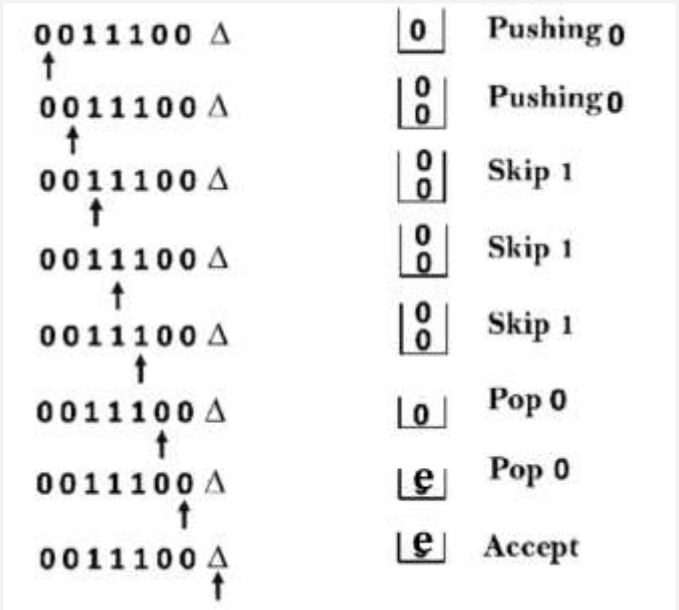
δ(q0, aaabbbbbb, Z) ⊢ δ(q0, aabbbbbb, aaZ)
⊢ δ(q0, abbbbbb, aaaaZ)
⊢ δ(q0, bbbbbb, aaaaaaZ)
⊢ δ(q1, bbbbb, aaaaaZ)
⊢ δ(q1, bbbb, aaaaZ)
⊢ δ(q1, bbb, aaaZ)
⊢ δ(q1, bb, aaZ)
⊢ δ(q1, b, aZ)
⊢ δ(q1, ε, Z)
⊢ δ(q2, ε)
ACCEPT

- Now we will simulate this PDA for the input string "aaabbbbbb".

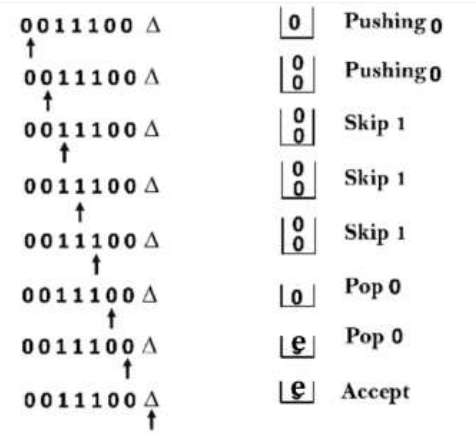
Example: Pushdown Automata(PDA)

- **Example-2 :** Design a PDA for accepting a language $\{0^n 1^m 0^n \mid m, n>=1\}$.

- **Solution:** In this PDA, n number of 0's are followed by any number of 1's followed n number of 0's. Hence the logic for design of such PDA will be as follows:
- Push all 0's onto the stack on encountering first 0's. Then if we read 1, just do nothing. Then read 0, and on each read of 0, pop one 0 from the stack.
- **For instance:**



Example: Pushdown Automata(PDA)



- **This scenario can be written in the ID form as:**

$\delta(q0, 0, Z) = \delta(q0, 0Z)$

$\delta(q0, 0, 0) = \delta(q0, 00)$

$\delta(q0, 1, 0) = \delta(q1, 0)$

$\delta(q0, 1, 0) = \delta(q1, 0)$

$\delta(q1, 0, 0) = \delta(q1, \varepsilon)$

$\delta(q0, \varepsilon, Z) = \delta(q2, Z)$ (ACCEPT state)

- Now we will simulate this PDA for the input string "0011100".

$\delta(q0, 0011100, Z) \vdash \delta(q0, 011100, 0Z)$

$\vdash \delta(q0, 11100, 00Z)$

$\vdash \delta(q0, 1100, 00Z)$

$\vdash \delta(q1, 100, 00Z)$

$\vdash \delta(q1, 00, 00Z)$

$\vdash \delta(q1, 0, 0Z)$

$\vdash \delta(q1, \varepsilon, Z)$

$\vdash \delta(q2, Z)$

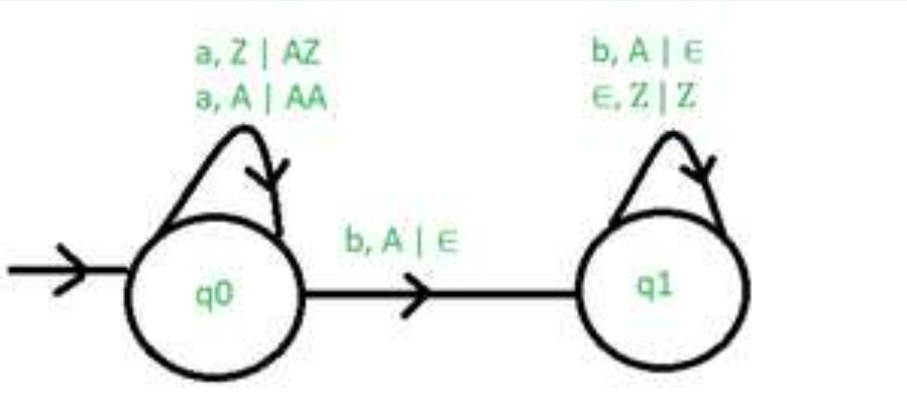
ACCEPT

Example: Pushdown Automata(PDA)

- **Example-3:** Define the pushdown automata for language $\{a^n b^n \mid n > 0\}$

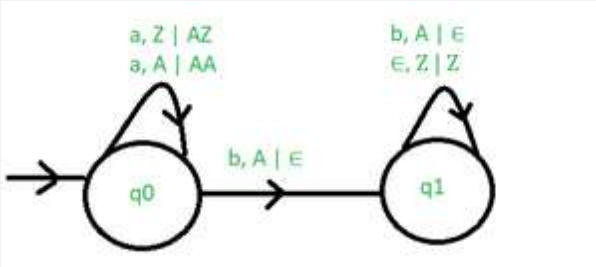
- **Solution :** Say, M = $\{\mathbf{Q, \sum, \Gamma, \delta, q_0, Z, F}\}$, where Q = { q0, q1 } and Σ = { a, b } and Γ = { A, Z } and δ is given by :

  δ(q0, a, Z) = { (q0, AZ) }
  δ(q0, a, A) = { (q0, AA) }
  δ(q0, b, A) = { (q1, ∈) }
  δ(q1, b, A) = { (q1, ∈) }
  δ(q1, ∈, Z) = { (q1, ∈) }



- Let us see how this automata works for the string “aaabbb”.

Example: Pushdown Automata(PDA)



- Let us see how this automata works for the string "aaabbb".

Row	State	Input	δ (transition function used)	Stack(Leftmost symbol represents top of stack)	State after move
1	q0	aaabbb		Z	q0
2	q0	**a**aabbb	δ(q0,a,Z)={(q0,AZ)}	AZ	q0
3	q0	a**a**abbb	δ(q0,a,A)={(q0,AA)}	AAZ	q0
4	q0	aa**a**bbb	δ(q0,a,A)={(q0,AA)}	AAAZ	q0
5	q0	aaa**b**bb	δ(q0,b,A)={(q1,∈)}	AAZ	q1
6	q1	aaab**b**b	δ(q1,b,A)= {(q1,∈)}	AZ	q1
7	q1	aaabb**b**	δ(q1,b,A)= {(q1,∈)}	Z	q1
8	q1	∈	δ(q1, ∈,Z)= {(q1,∈)}	∈	q1

Explanation:

- Initially, the state of automata is q0 and symbol on stack is Z and the input is aaabbb as shown in row 1.
- On reading 'a' (shown in bold in row 2), the state will remain q0 and it will push symbol A on stack.
- On next 'a' (shown in row 3), it will push another symbol A on stack.
- After reading 3 a's, the stack will be AAAZ with A on the top.
- After reading 'b' (as shown in row 5), it will pop A and move to state q1 and stack will be AAZ.
- When all b's are read, the state will be q1 and stack will be Z.
- In row 8, on input symbol '∈' and Z on stack, it will pop Z and stack will be empty.
- This type of acceptance is known as **acceptance by empty stack.**

?

THE END