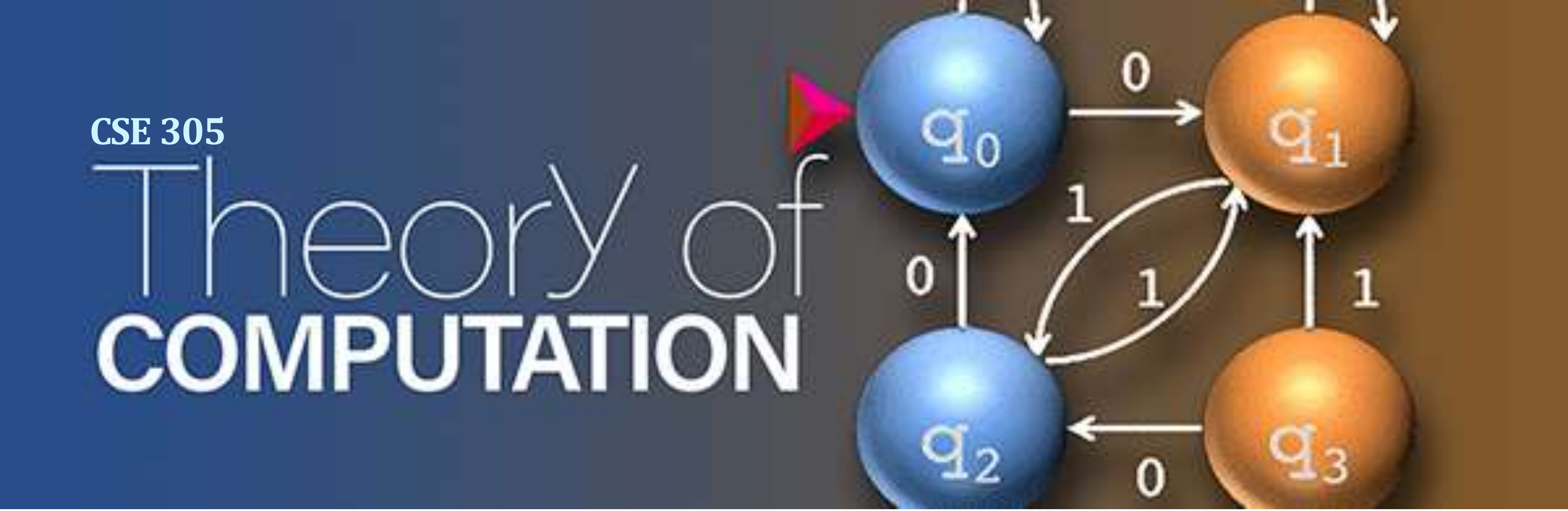


Lecture 27

Pushdown Automata (2)

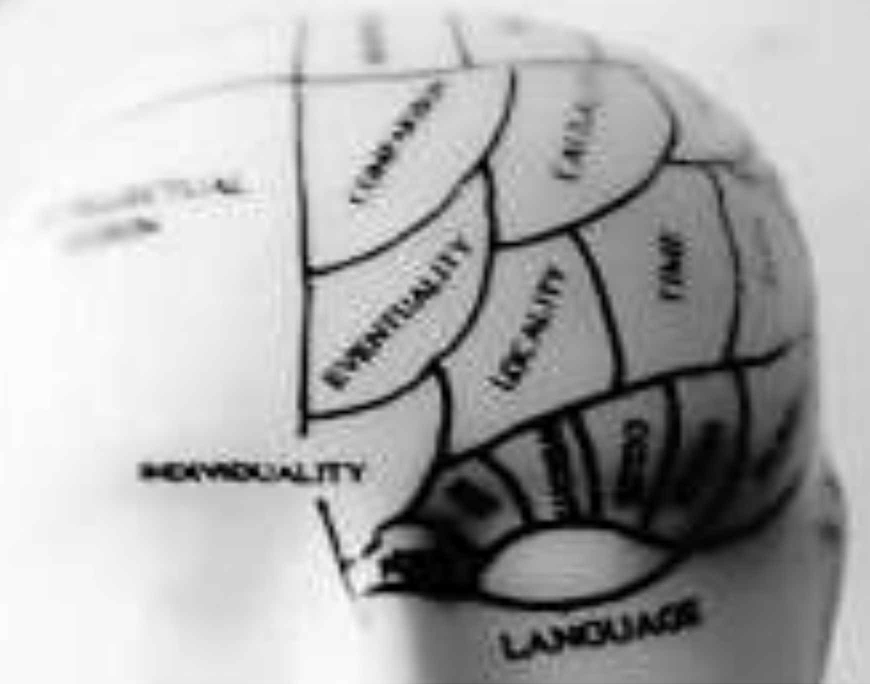
Md. Mijanur Rahman, Prof. Dr.

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

PDA Acceptance

- A language can be accepted by Pushdown automata using two approaches:

1. **Acceptance by Final State:** The PDA is said to accept its input by the final state if it enters any final state in zero or more moves after reading the entire input.

2. **Acceptance by Empty Stack:** On reading the input string from the initial configuration for some PDA, the stack of PDA gets empty.

PDA Acceptance

- **Acceptance by Final State:**

  Let P =(Q, ∑, Γ, δ, q0, Z, F) be a PDA. The language acceptable by the final state can be defined as:

  L(PDA) = {w | (q0, w, Z) ⊢* (p, ε, ε), q ∈ F}

- If there is a language L = L (P1) for some PDA P1 then there is a PDA P2 such that L = N(P2). That means language accepted by final state PDA is also acceptable by empty stack PDA.

PDA Acceptance

- **Acceptance by Empty Stack:**

  Let P =(Q, ∑, Γ, δ, q0, Z, F) be a PDA. The language acceptable by empty stack can be defined as:

  $N(PDA) = \{w \mid (q0, w, Z) \vdash^* (p, \varepsilon, \varepsilon), q \in Q\}$

- If L = N(P1) for some PDA P1, then there is a PDA P2 such that L = L(P2). That means the language accepted by empty stack PDA will also be accepted by final state PDA.

Example

- **Example:** Construct a PDA that accepts the language L over {0, 1} by empty stack which accepts all the string of 0's and 1's in which a number of 0's are twice of number of 1's.
- **Solution:**
- There are two parts for designing this PDA:
  - If 1 comes before any 0's
  - If 0 comes before any 1's.
- We are going to design the first part i.e. 1 comes before 0's. The logic is that read single 1 and push two 1's onto the stack. Thereafter on reading two 0's, POP two 1's from the stack. The δ can be

  δ(q0, 1, Z) = (q0, 11, Z)   Here Z represents that stack is empty

  δ(q0, 0, 1) = (q0, ε)

Example

- Now, consider the second part i.e. if 0 comes before 1's. The logic is that read first 0, push it onto the stack and change state from q0 to q1. [Note that state q1 indicates that first 0 is read and still second 0 has yet to read].
- Being in q1, if 1 is encountered then POP 0. Being in q1, if 0 is read then simply read that second 0 and move ahead. The δ will be:

  δ(q0, 0, Z) = (q1, 0Z)

  δ(q1, 0, 0) = (q1, 0)

  δ(q1, 0, Z) = (q0, ε)

  (indicate that one 0 and one 1 is already read, so simp ly read the second 0)

  δ(q1, 1, 0) = (q1, ε)

- Now, summarize the complete PDA for given L is:

  δ(q0, 1, Z) = (q0, 11Z)

  δ(q0, 0, 1) = (q1, ε)

  δ(q0, 0, Z) = (q1, 0Z)

  δ(q1, 0, 0) = (q1, 0)

  δ(q1, 0, Z) = (q0, ε)

  δ(q0, ε, Z) = (q0, ε)   ACCEPT state

Non-deterministic PDA

- The non-deterministic pushdown automata is very much similar to NFA. We will discuss some CFGs which accepts NPDA.
- The CFG which accepts deterministic PDA accepts non-deterministic PDAs as well.
- Similarly, there are some CFGs which can be accepted only by NPDA and not by DPDA.
- Thus NPDA is more powerful than DPDA.

Non-deterministic PDA

- **Example: Design PDA for Palindrome strips.**

Solution:

- Suppose the language consists of string L = {aba, aa, bb, bab, bbabb, aabaa, ...]. The string can be odd palindrome or even palindrome.
- The logic for constructing PDA is that we will push a symbol onto the stack till half of the string then we will read each symbol and then perform the pop operation.
- We will compare to see whether the symbol which is popped is similar to the symbol which is read.
- Whether we reach to end of the input, we expect the stack to be empty.

Non-deterministic PDA

- This PDA is a non-deterministic PDA because finding the mid for the given string and reading the string from left and matching it with from right (reverse) direction leads to non-deterministic moves. Here is the ID.

1. δ(q1, a, Z) = (q1, aZ)
2. δ(q0, b, Z) = (q1, bZ)
3. δ(q0, a, a) = (q1, aa)
4. δ(q1, a, b) = (q1, ab)
5. δ(q1, a, b) = (q1, ba)
6. δ(q1, b, b) = (q1, bb)

Pushing the symbols onto the stack

7. δ(q1, a, a) = (q2, ε)
8. δ(q1, b, b) = (q2, ε)
9. δ(q2, a, a) = (q2, ε)
10. δ(q2, b, b) = (q2, ε)
11. δ(q2, ε, Z) = (q2, ε)

Popping the symbols on reading the same kind of symbol

- **Simulation of the string "abaaba":**

δ(q1, abaaba, Z)	Apply rule 1
⊢ δ(q1, baaba, aZ)	Apply rule 5
⊢ δ(q1, aaba, baZ)	Apply rule 4
⊢ δ(q1, aba, abaZ)	Apply rule 7
⊢ δ(q2, ba, baZ)	Apply rule 8
⊢ δ(q2, a, aZ)	Apply rule 7
⊢ δ(q2, ε, Z)	Apply rule 11
⊢ δ(q2, ε)	Accept

PDA & CFG

- 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)

PDA & CFG

Algorithm to find PDA corresponding to a given CFG

- **Input** − A CFG, G = (V, T, P, S)
- **Output** − Equivalent PDA, P = (Q, ∑, S, δ, q_0, I, F)
- Procedure:

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)**.

PDA & CFG

- **Example:** Convert the following grammar to a PDA that accepts the same language.

  $S \rightarrow 0S1 \mid A$

  $A \rightarrow 1A0 \mid S \mid \varepsilon$

- **Solution:** The CFG can be first simplified by eliminating unit productions:

  $S \rightarrow 0S1 \mid 1S0 \mid \varepsilon$

- Now we will convert this CFG to GNF:

  $S \rightarrow 0SX \mid 1SY \mid \varepsilon$

  $X \rightarrow 1$

  $Y \rightarrow 0$

- **The PDA can be:**

R1: δ(q, ε, S) = {(q, 0SX) | (q, 1SY) | (q, ε)}

R2: δ(q, ε, X) = {(q, 1)}

R3: δ(q, ε, Y) = {(q, 0)}

R4: δ(q, 0, 0) = {(q, ε)}

R5: δ(q, 1, 1) = {(q, ε)}

PDA & CFG

- **Example: Construct PDA for the given CFG, and test whether 010^4 is acceptable by this PDA.**

  S → 0BB

  B → 0S | 1S | 0

- **Solution:** The PDA can be given as:

  A = {(q), (0, 1), (S, B, 0, 1), δ, q, S, ?}

  The production rule δ can be:

```
R1: δ(q, ε, S) = {(q, 0BB)}
R2: δ(q, ε, B) = {(q, 0S) | (q, 1S) | (q, 0)}
R3: δ(q, 0, 0) = {(q, ε)}
R4: δ(q, 1, 1) = {(q, ε)}
```

- Testing 010^4 i.e. 010000 against PDA:

```
δ(q, 010000, S) ⊢ δ(q, 010000, 0BB)
        ⊢ δ(q, 10000, BB)      R1
        ⊢ δ(q, 10000,1SB)      R3
        ⊢ δ(q, 0000, SB)       R2
        ⊢ δ(q, 0000, 0BBB)     R1
        ⊢ δ(q, 000, BBB)       R3
        ⊢ δ(q, 000, 0BB)       R2
        ⊢ δ(q, 00, BB)         R3
        ⊢ δ(q, 00, 0B)         R2
        ⊢ δ(q, 0, B)           R3
        ⊢ δ(q, 0, 0)           R2
        ⊢ δ(q, ε)              R3
        ACCEPT
```

- Thus 010^4 is accepted by the PDA.

THE END