CSE211 – Formal Languages and Automata Theory

U2L5_Push Down Automata (PDA)

Dr. P. Saravanan
School of Computing
SASTRA Deemed University

Agenda

- Recap of previous class
- Introduction
- Definition of PDA
- The Language of a PDA
- Equivalence of PDA's and CFG's
- Deterministic PDA's

Recap of previous class

- CFL
- Derivation
- Parse Tree
- Ambiguous grammar
- Removing ambiguity

PDA - Introduction

Basic concepts:

- CFL's may be accepted by pushdown automata (PDA's)
- A PDA is an e-NFA with a stack
- The stack can be read, pushed, and popped only on the top
- Two different versions of PDA's:
  - Accepting strings by "entering an accepting state";
  - Accepting strings by "emptying the stack."

PDA - Introduction

Basic concepts:

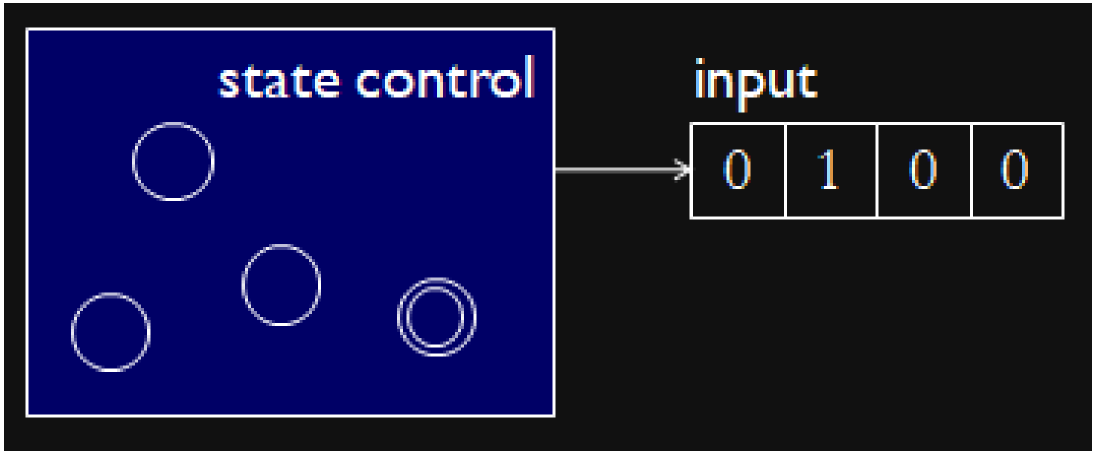
- The original PDA is *nondeterministic.*
- There is also a subclass of PDA's which are *deterministic* in nature.
- Deterministic PDA's (DPDA's) resembles parsers for CFL's in compilers.
- It is interesting to know what "language constructs" which a DPDA can accept.
- The stack is *infinite* in size, so can be used as a "memory" to eliminate the weakness of "finite states" of NFA's, which cannot accept languages like $L = \{a^n b^n \mid n \geq 1\}$.

PDA - Introduction

- Advantage of the stack --- the stack can "remember" an *infinite* amount of information.
- Weakness of the stack --- the stack can only be read in a *first-in-last-out* manner.
- Therefore, it can accept languages like $L_{ww^r} = \{ww^R \mid w$ is in $(\mathbf{0} + \mathbf{1})^*\}$, but not languages like $L = \{a^n b^n c^n \mid n \geq 1\}$.

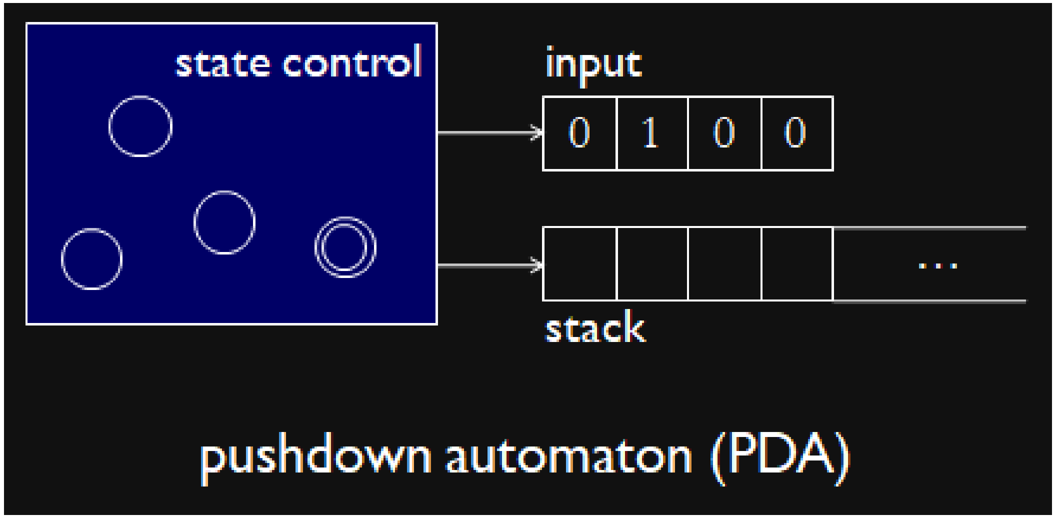
Push Down Automata Vs NFA

- Since context-free is more powerful than regular, pushdown automata must generalize NFAs



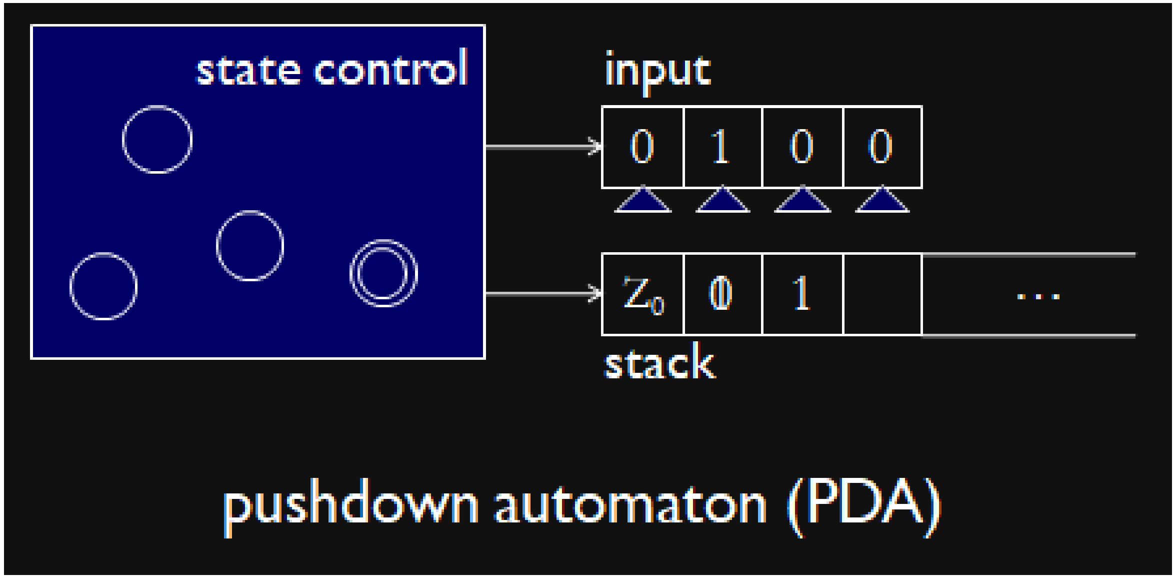
Push Down Automata Vs NFA

- A pushdown automaton has access to a stack, which is a potentially infinite supply of memory



Push Down Automata Vs NFA

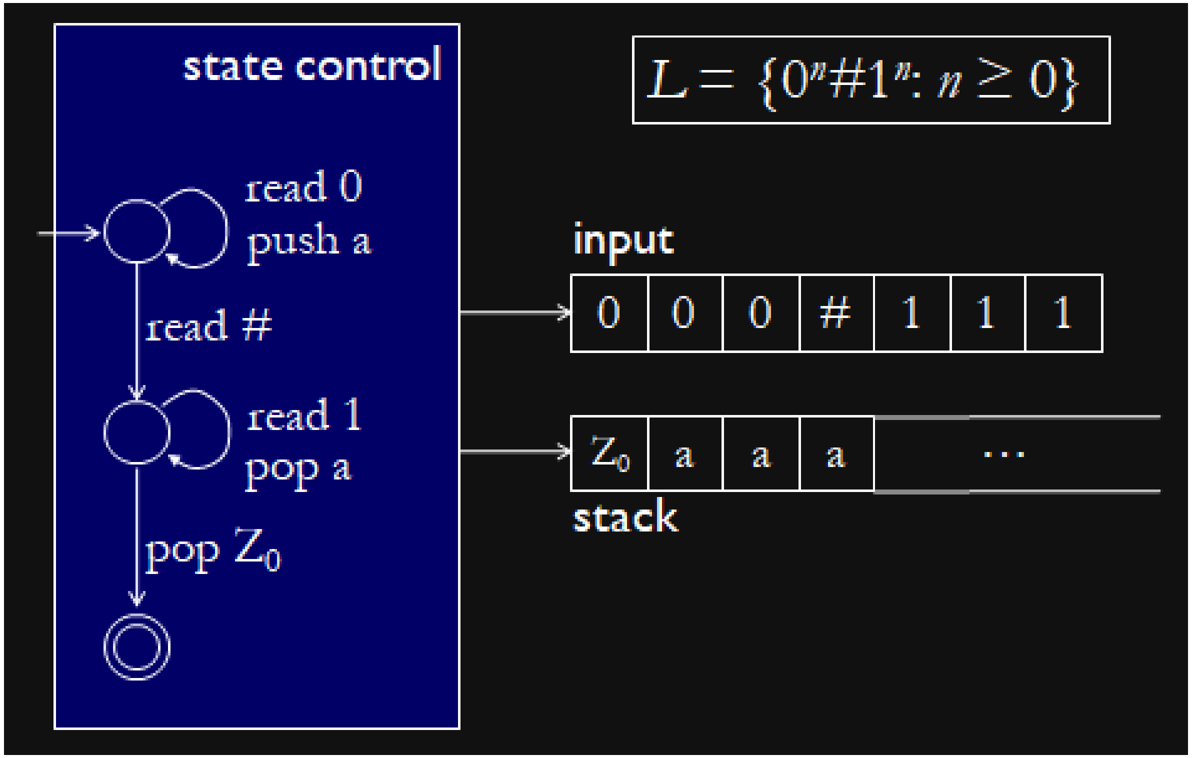
- As the PDA is reading the input, it can push / pop symbols in / out of the stack



Rules for pushdown automata

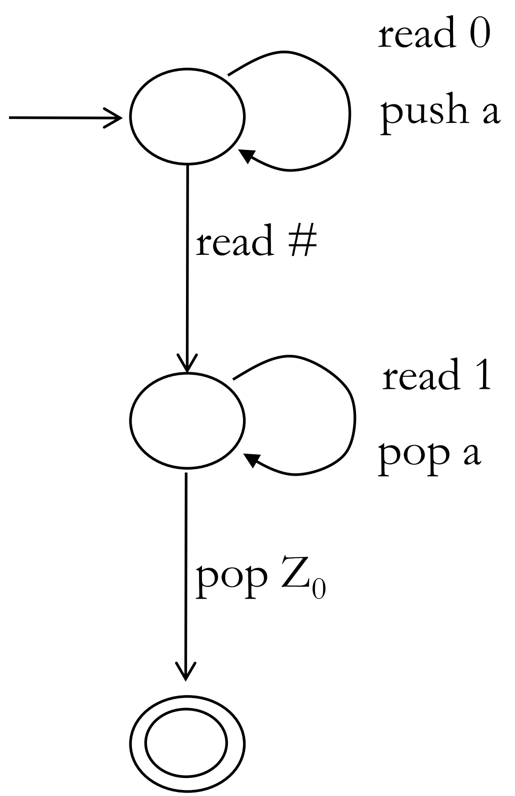
- The transitions are nondeterministic
- Stack is always accessed from the top
- Each transition can pop a symbol from the stack and / or push another symbol onto the stack
- Transitions depend on input symbol and on last symbol popped from stack
- Automaton accepts if after reading whole input, it can reach an accepting state

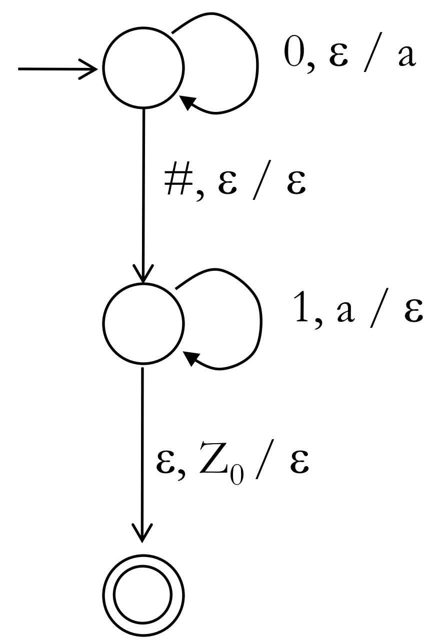
Example



state control
read 0
push a
read #
read 1
pop a
pop Z_0
$L = \{0^n\#1^n: n \geq 0\}$
input
0 0 0 # 1 1 1
Z_0 a a a ...
stack

Shorthand notation

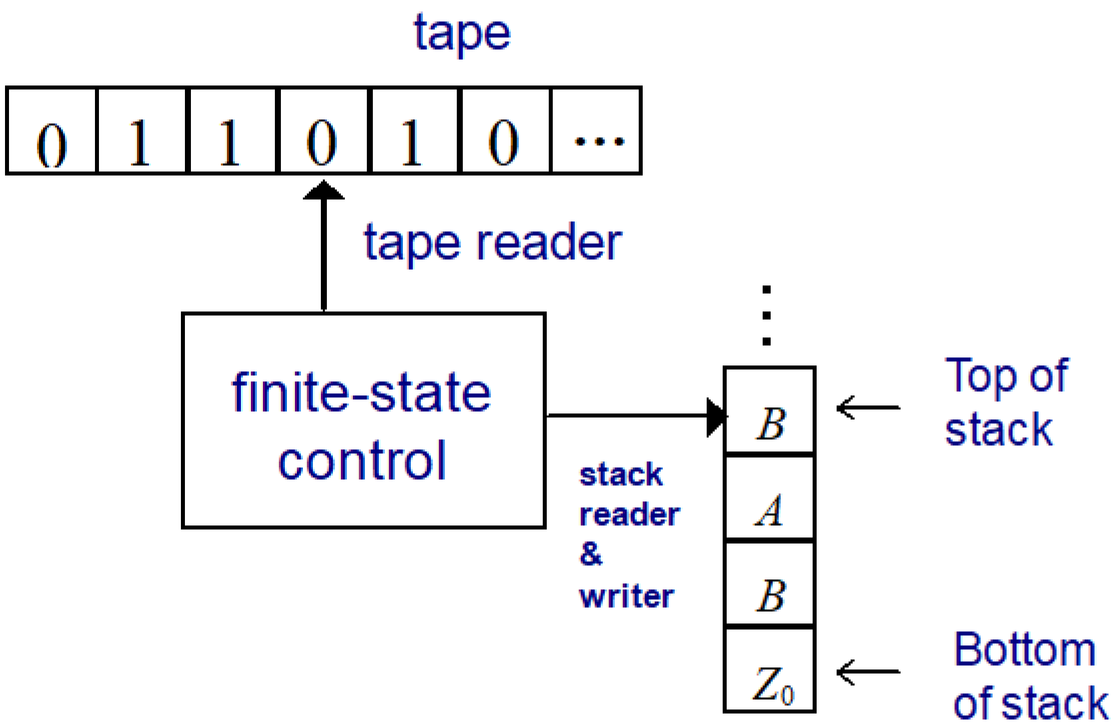




read, pop / push

Graphical Model of PDA

- A graphic model of a PDA



A graph model of a PDA

Formal Definition

A PDA is a 7-tuple $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where

- Q: a finite set of states
- Σ: a finite set of input symbols
- Γ: a finite stack alphabet
- δ: a transition function such that $\delta(q, a, X)$ **is a set of pairs** (p, γ) where
  - $q \in Q$ (the current state)
  - $a \in \Sigma$ or $a = \varepsilon$ (an input symbol or an empty string)
  - $X \in \Gamma$
  - $p \in Q$ (the next state)

Formal Definition…

- $\gamma \in \Gamma^*$ which replaces X on the top of the stack:

  when $\gamma = \varepsilon$, the top stack symbol is popped up

  when $\gamma = X$, the stack is unchanged

  when $\gamma = YZ$, X is replaced by Z, and Y is pushed to the top

  when $\gamma = \alpha Z$, X is replaced by Z and string α is pushed to the top
- q_0: the start state
- Z_0: the start symbol of the stack
- F: the set of accepting or final states

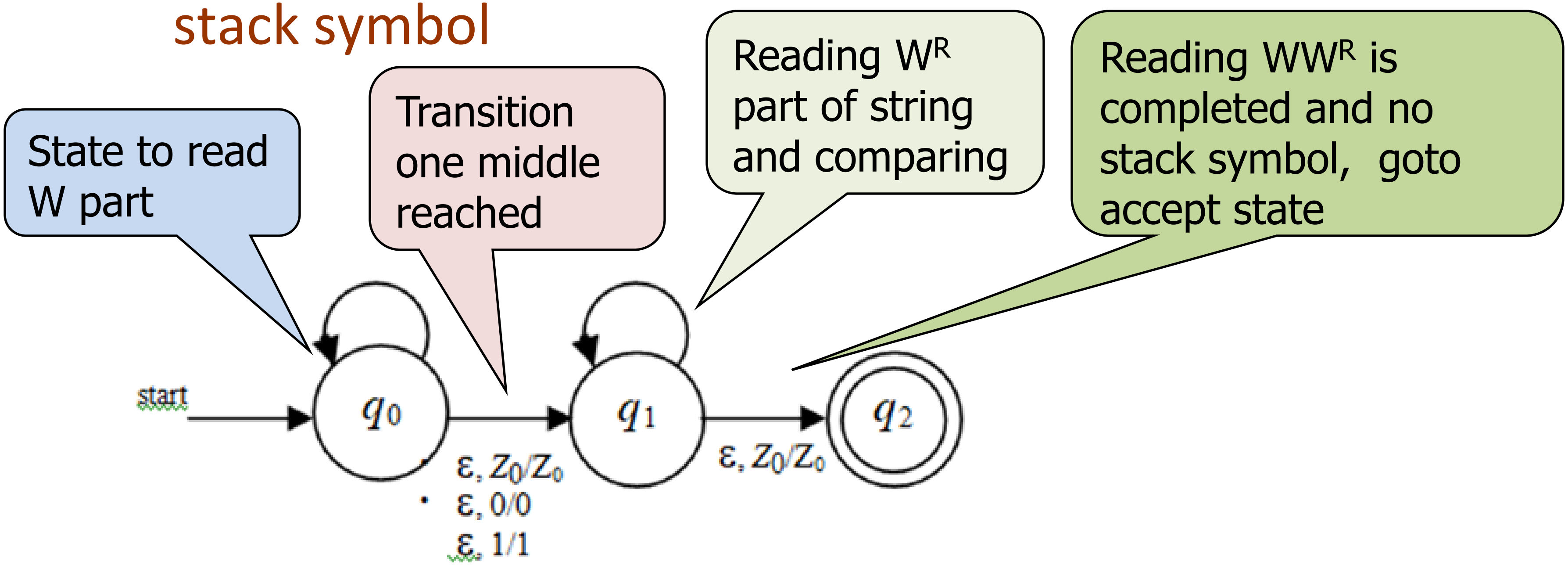
Designing PDA means defining all these elements

Designing PDA: Example

- Example: 6.1 - Design a PDA to accept the language $L_{wwr} = \{ww^R \mid w \text{ is in } (0 + 1)^*\}$
- In start state q0, copy input symbols onto the stack
- At any time, nondeterministically guess whether the middle of ww^R is reached and enter q1, or continue copying input symbols.
- In q1, compare remaining input symbols with those on the stack one by one.
- If the stack can be so emptied, then the matching of w with w^R succeeds.

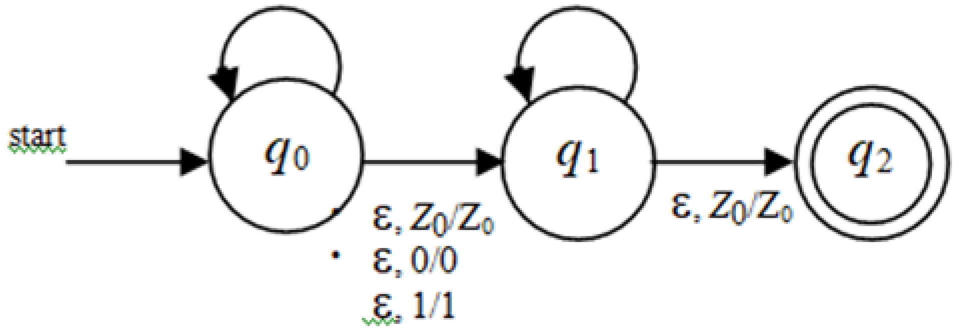
Designing PDA Example

- Designing a PDA to accept the language $L_{ww}{}^{R}$. Where $\Sigma=\{0,1\}$ and $\Gamma=\{a,b\}$
  - With stack symbol – use a for 0 and use b for 1
  - Without stack symbol – use 0 for 0 and use 1 for 1 as stack symbol



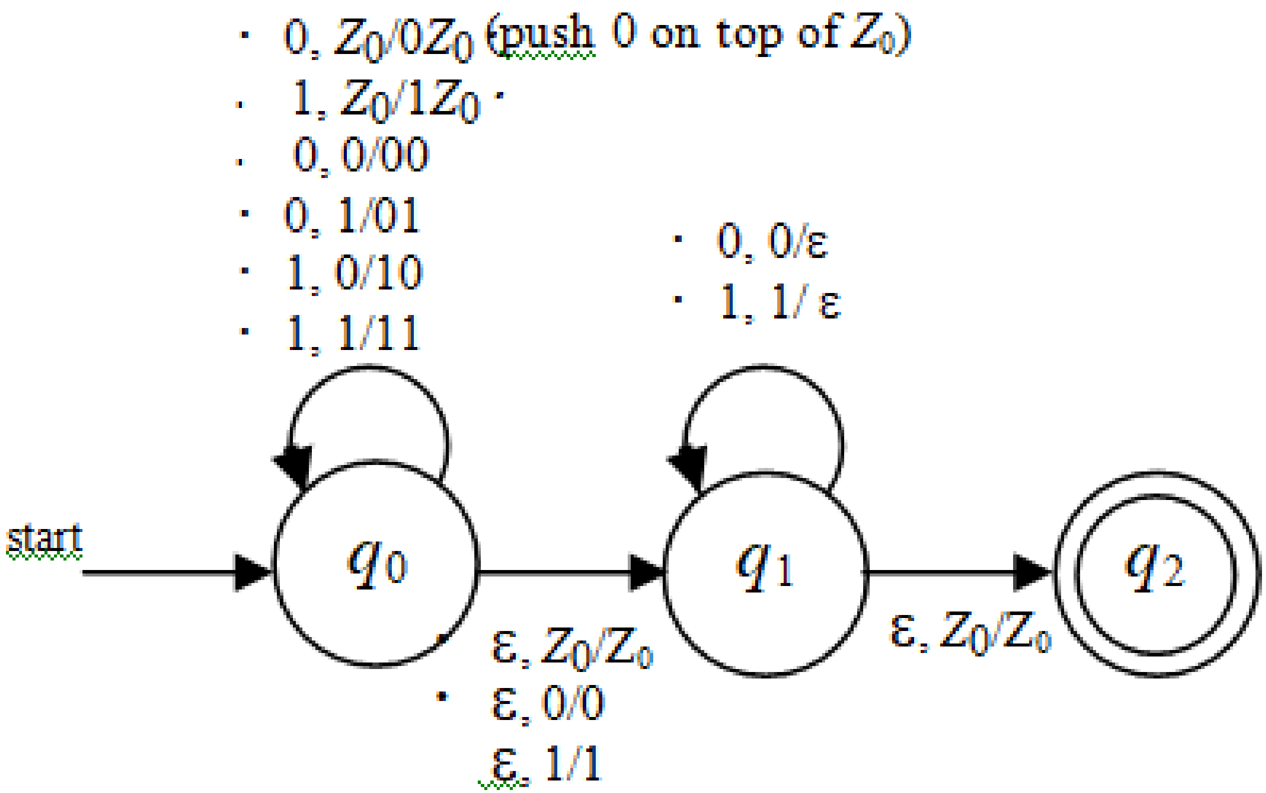
Designing PDA Example

- 10100101



Designing PDA Example

- Designing a PDA to accept the language L_{ww^R}.



Designing PDA: Example

- Designing a PDA to accept the language L_{ww^R}.
  - Need a start symbol Z of the stack and a 3rd state q_2 as the accepting state.
  - $P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\})$ such that
    - $\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$, $\delta(q_0, 1, Z_0) = \{(q_0, 1Z_0)\}$

      (initial pushing steps with Z_0 to mark stack bottom)
    - $\delta(q_0, 0, 0) = \{(q_0, 00)\}$, $\delta(q_0, 0, 1) = \{(q_0, 01)\}$,

      $\delta(q_0, 1, 0) = \{(q_0, 10)\}$, $\delta(q_0, 1, 1) = \{(q_0, 11)\}$

Rules for pushdown automata

- $\delta(q_0, \varepsilon, Z_0) = \{(q_1, Z_0)\}$

  (check if input is ε which is in L_{ww^R})

- $\delta(q_0, \varepsilon, 0) = \{(q_1, 0)\}$, $\delta(q_0, \varepsilon, 1) = \{(q_1, 1)\}$

  (check the string's middle)

- $\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\}$, $\delta(q_1, 1, 1) = \{(q_1, \varepsilon)\}$

  (matching pairs)

- $\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$

  (entering final state)

Summary

- Definition of PDA
- Designing of PDA

References

- John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman, *Introduction to Automata Theory,* Languages, and Computation, Pearson, 3rd Edition, 2011.
- Peter Linz, An Introduction to Formal Languages and Automata, Jones and Bartle Learning International, United Kingdom, 6th Edition, 2016.

Next Class:

ID, Language, Equivalence

THANK YOU.