VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

BCSE304L
Theory of Computation

M5L1 – Introduction of PDA

Dr. P. Saravanan

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:

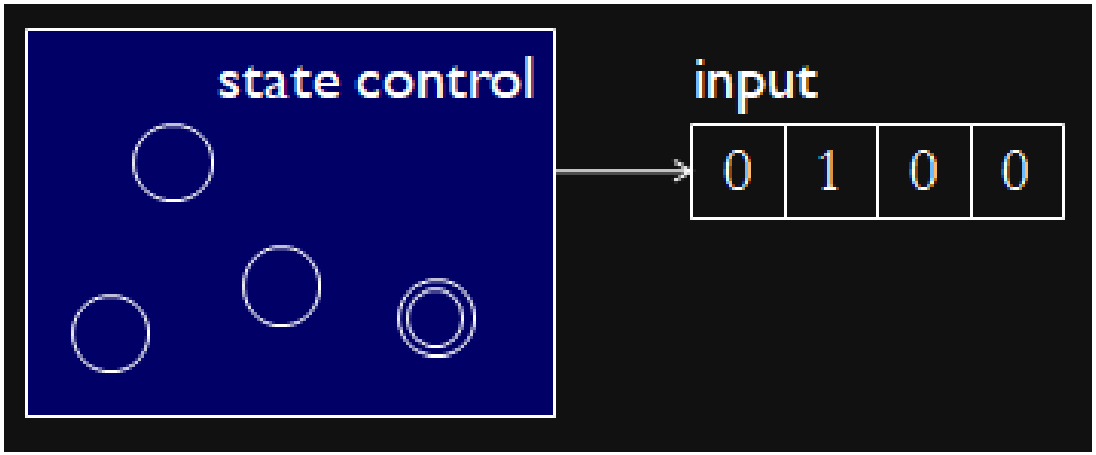
- The original PDA is *nondeterministic.*
- There is also a subclass of PDA's which are *deterministic* in nature.
- Deterministic PDA's (DPDA's) resembles parsers for CFL's in compilers.
- It is interesting to know what "language constructs" which a DPDA can accept.
- The stack is *infinite* in size, 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 $L = \{a^{n}b^{n}c^{n} \mid n \geq 1\}$.

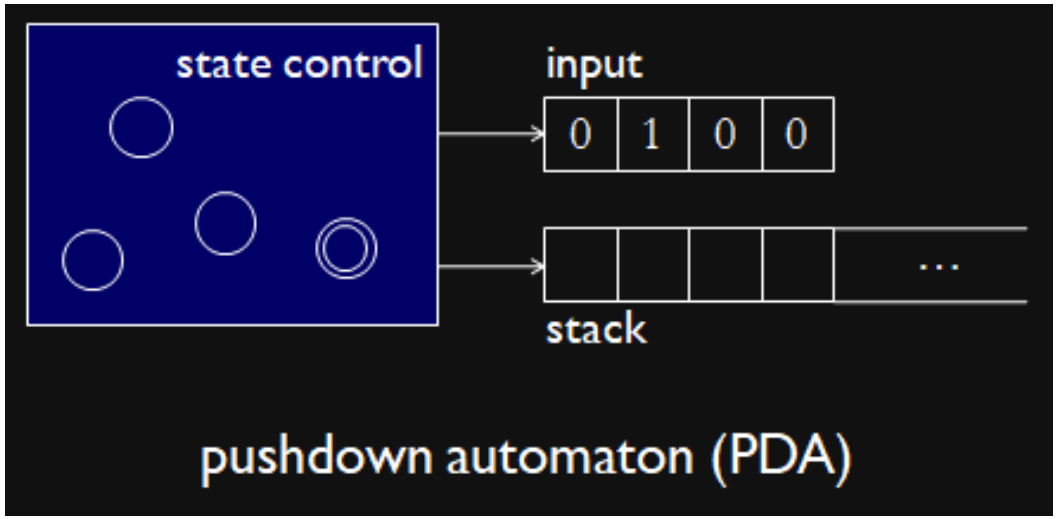
Push Down Automata Vs NFA

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



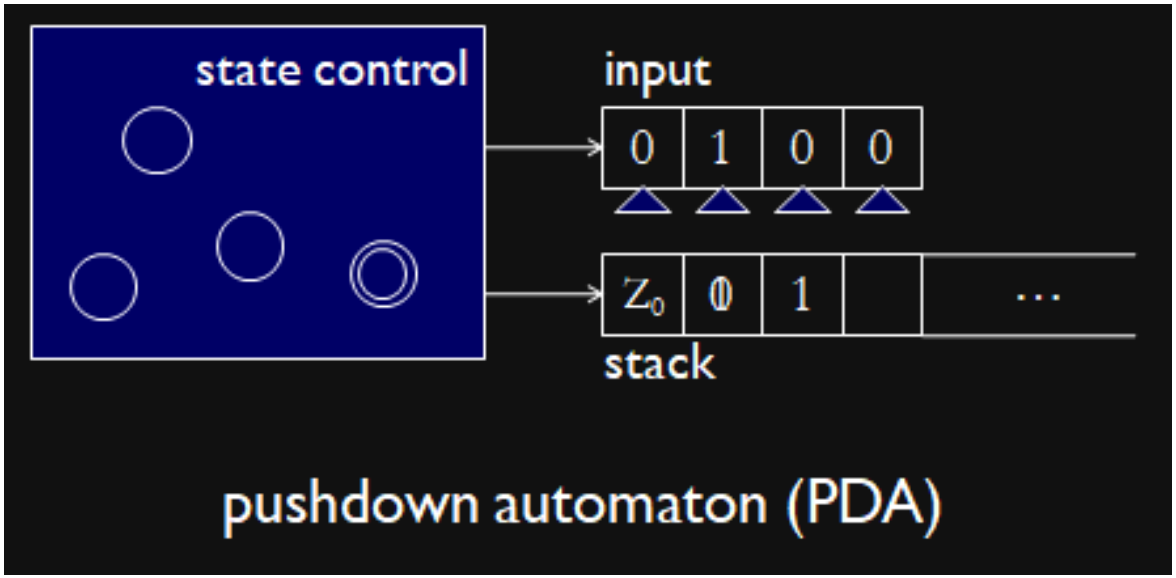
Push Down Automata Vs NFA

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



Push Down Automata Vs NFA

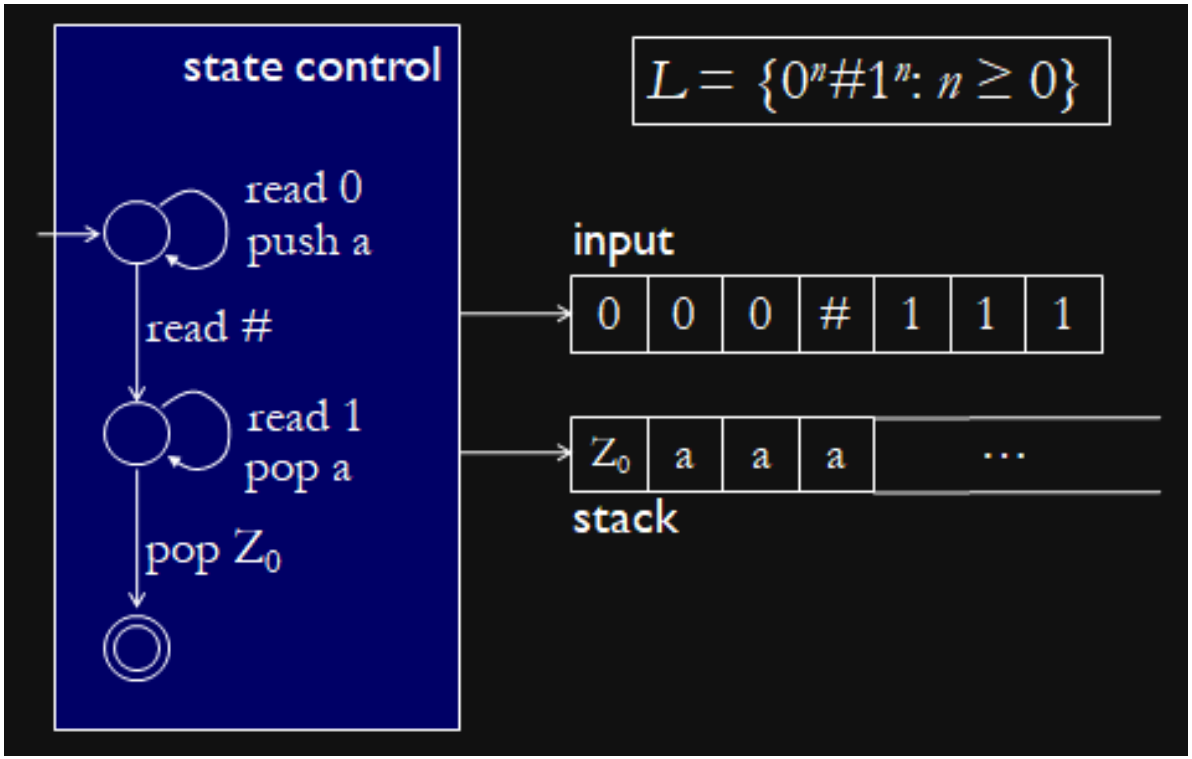
- As the PDA is reading the input, it can push / pop symbols in / out of the stack



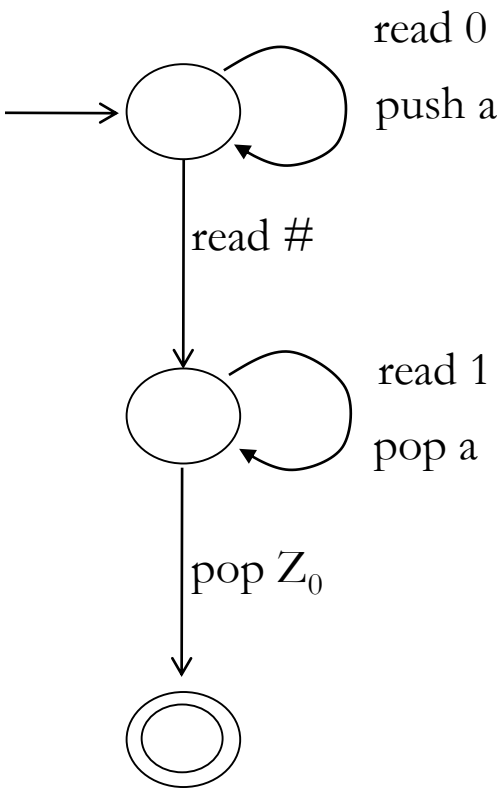
Rules for pushdown automata

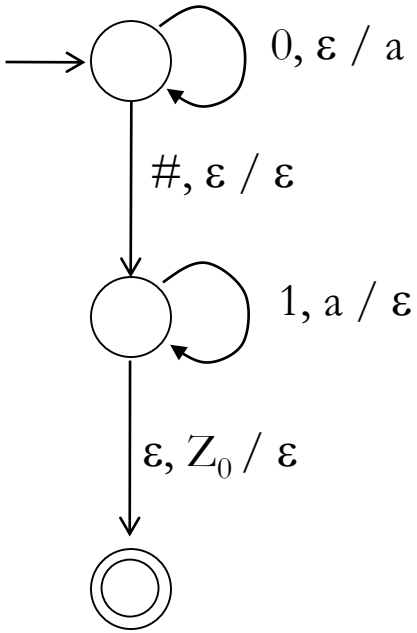
- The transitions are nondeterministic
- Stack is always accessed from the top
- Each transition can pop a symbol from the stack and / or push another symbol onto the stack
- Transitions depend on input symbol and on last symbol popped from stack
- Automaton accepts if after reading whole input, it can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

Example


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

Shorthand notation

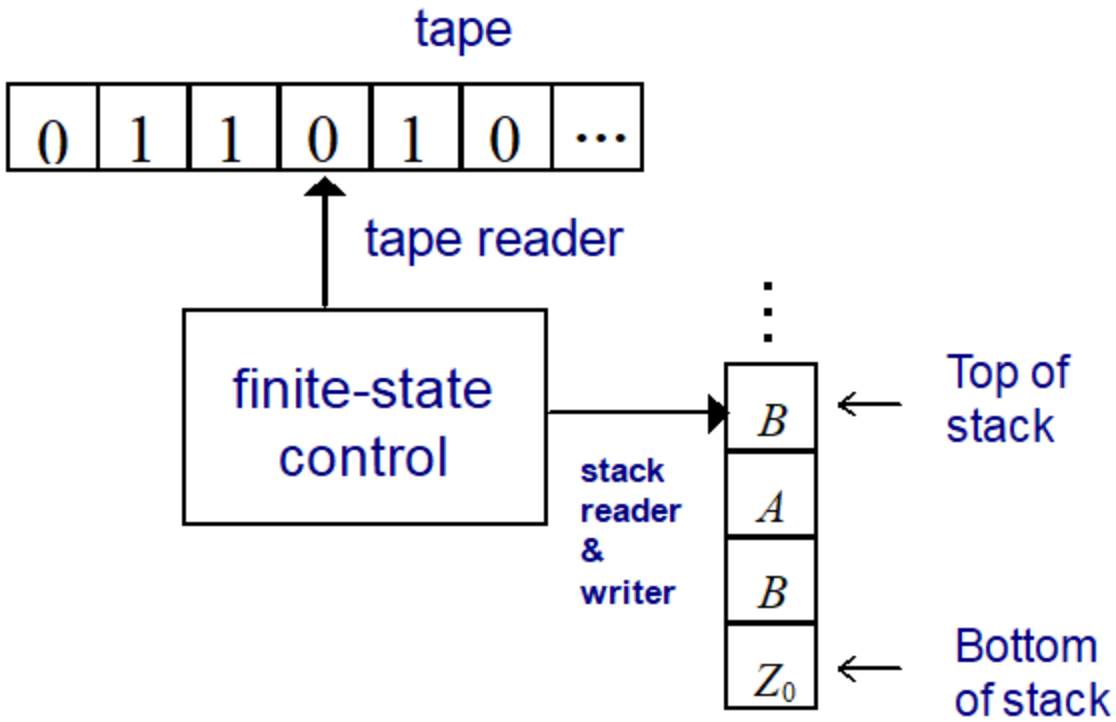




read, pop / push

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

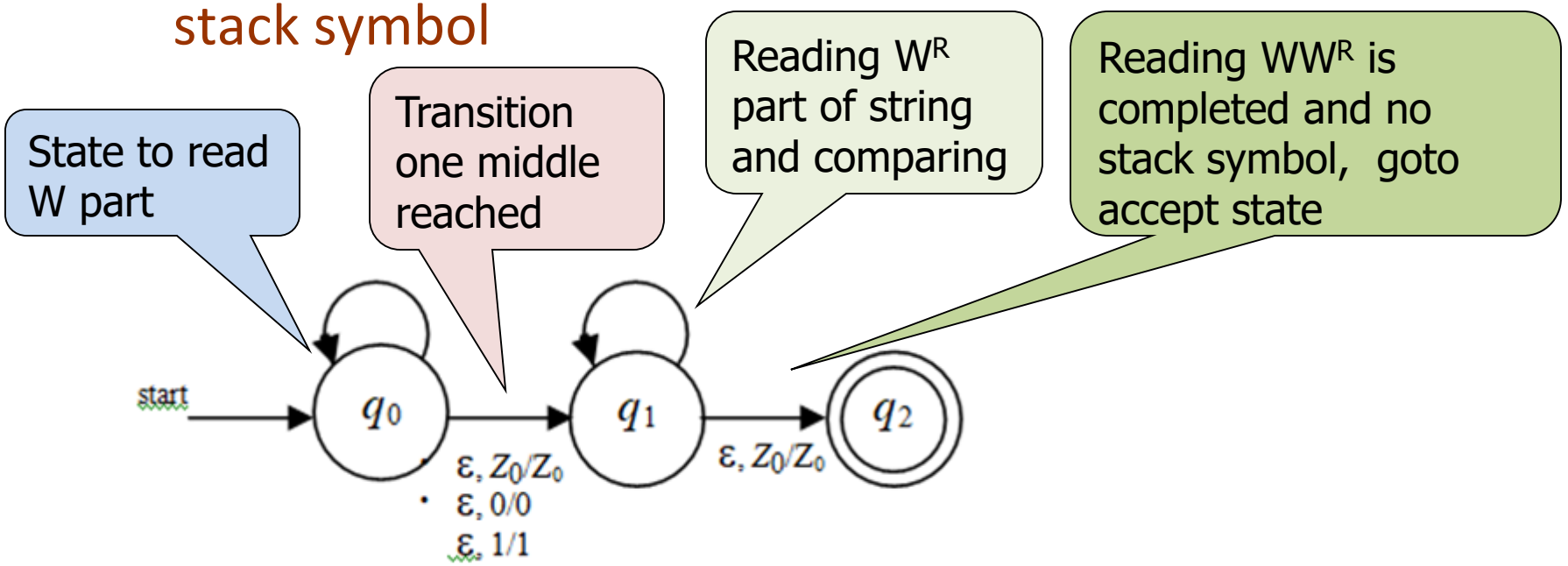
Designing PDA means defining all these elements

Designing PDA: Example

- Example: 6.1 - Design a PDA to accept the language Lwwr = {ww^R | w 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.

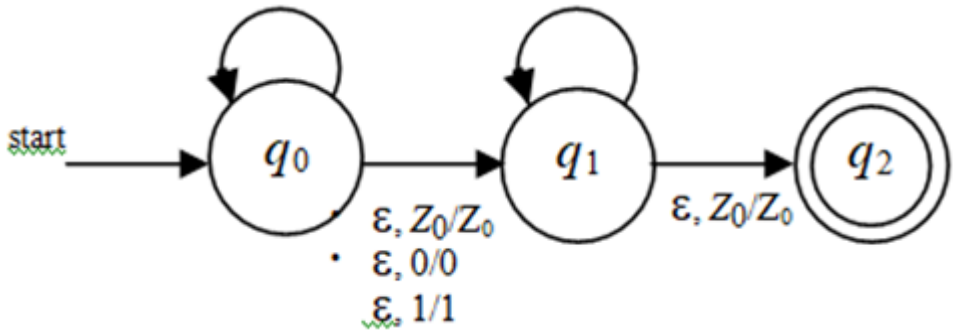
Designing PDA Example

- 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



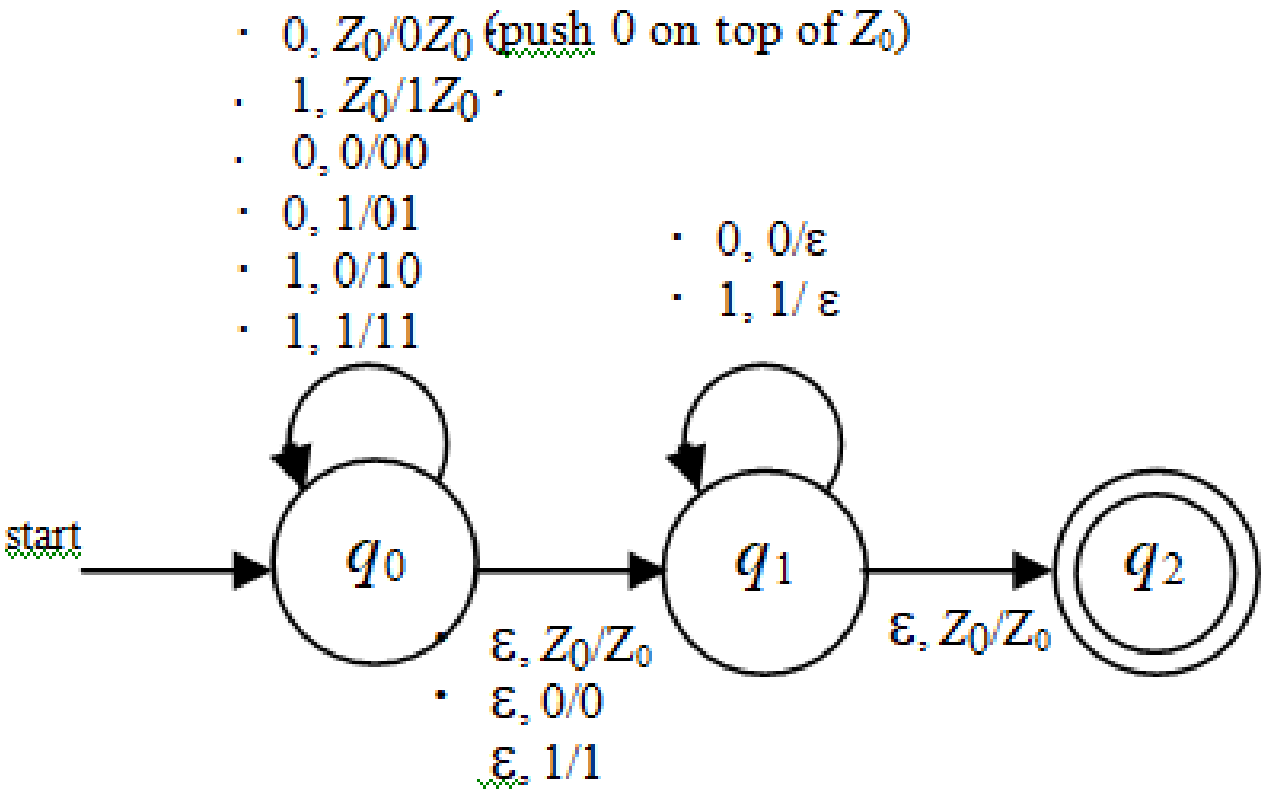
Designing PDA Example

- 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.

Pumping Lemma for CFL

- **Theorem** (pumping lemma for CFL's)
  - Let L be a CFL. There exists an integer constant n such that if $z \in L$ with $|z| \geq n$, then we can write $z = uvwxy$, subject to the following conditions:
    1. $|vwx| \leq n$;
    2. $v, x \neq \varepsilon$ (that is, v, x are not both ε);
    3. for all $i \geq 0$, $uv^iwx^iy \in L$.
- Used to prove that the given language is not a context-free language

Example 1

- Prove by contradiction the language $L = \{0^m 1^m 2^m \mid m \geq 1\}$ is not a CFL by the pumping lemma.
- *Proof*.
  - Step 1: an integer constant $n=9$
  - Step 2: such that if $z \in L$ with $|z| \geq 9$, then we can write $z = 000111222$,
  - Step 3: Divide the string z into $z = uvwxy$ *such that*

    $|vwx| \leq n$; and $v, x \neq \varepsilon$ (that is, v, x are not both ε);

    $z = 00 \quad 01 \quad 11 \quad 2 \quad 22$

    U V W X Y
  - Step 4: for all $i \geq 0$, $uv^i wx^i y \in L$

For i=0 => uv^0wx^0y => 001122 $\in L$

For i=1 => uv^1wx^1y

=> 000111222 $\in L$

For i=2 => uv^2wx^2y

=> 00 0101 11 22 22 $\notin L$

So the given language is not a CFL

Example 2:

- Prove that $L=\{ww \mid w \in \{0, 1\}^*\}$ is not a CFL.

Closure Properties of CFL's

- **Some differences between CFL's and RL's**
  - CFL's are *not* closed under *intersection*, *difference*, or *complementation*
  - But the intersection or difference of a CFL and an RL is still a CFL.
  - We will introduce a new operation --- substitution.

Substitution

- **Definitions:**
  - A *substitution s* on an alphabet S is a function such that for each $a \in S$, $s(a)$ is a language L_a over any alphabet (not necessarily S)
  - For a string $w = a_1a_2...a_n \in S^*$, $s(w) = s(a_1)s(a_2)...s(a_n) = L_{a1}L_{a2}...L_{an}$, i.e., $s(w)$ is a language which is the concatenation of all L_{ai}'s
  - Given a language L, $s(L) = \cup_{w \in L} s(w)$

Substitution Example

- A *substitution s* on an alphabet

  S = {0, 1} is defined as

  $S(0) = \{a^n b^n \mid n \geq 1\}$, $s(1) = \{aa, bb\}$.

- Let w = 01, then
  - $s(w) = s(0)s(1)$
  - $= \{a^n b^n \mid n \geq 1\}\{aa, bb\}$
  - $= \{a^n b^n aa \mid n \geq 1\} \cup \{a^n b^{n+2} \mid n \geq 1\}$.

Closure properties of CFL

The CFL's are closed under the following operations:

- 1. Union
- 2. Concatenation
- 3. Closure (*), and positive closure (+)
- 4. Homomorphism
- 5. Inverse Homomorphism
- 6. Reversal

Not Closed

- 1. Intersection
- 2. Difference and
- 3. Complementation

Intersection with an RL

- **Theorem 7.27**
  - If L is a CFL and R is an RL, then $L \cap R$ is a CFL.
- The following are true about CFL's L, L_1, and L_2, and an RL R:
- 1. $L - R$ is a CFL;
- 2. $\overline{L}$ is *not* necessarily a CFL;
- 3. $L_1 - L_2$ is *not* necessarily a CFL.

Decision Properties of CFL's

- **Facts:**
  - Unlike RLs' decision problems which are all solvable, *very little* can be said about CFL's.
  - Only two problems *can* be *decided* for CFL's:
    - whether the language is empty;
    - whether a given string is in the language.
  - Computational complexity for conversions between CFG's and PDA's will be investigated.

Decision Properties of CFL's

- **Testing Emptiness of CFL's**
- The problem of testing emptiness of a CFL *L* is *decidable*.
- **Testing Membership in a CFL**
- A way for solving the membership problem for a CFL *L* is to use the CNF of the CFG *G* for *L* in the following way:
  - The parse tree of an input string *w* of length *n* using the CNF grammar *G* has $2n - 1$ nodes.
  - We can generate all possible parse trees and check if a yield of them is *w*.
- The number of such trees is *exponential* in *n*.

Preview of Un-decidable CFL Problems

- **The following are undecidable CFL problems ---**
  - Is a given CFL inherently ambiguous?
  - Is the intersection of two CFL's empty?
  - Are two CFL's the same?
  - Is a given CFL equal to S*, where S is the alphabet of this language?

Summary

- Recap of previous class
  - Normal Forms
- Pumping Lemma for CFL
  - Definition
  - Examples
- Closure Properties
- Decision Properties

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.