Damascus University

PushDown Automata

PDA- Part II

Properties of Context-free Languages

Dr. Mohammad Ahmad

Pushdown Automata - Definition

- A PDA P := (Q,∑,Γ, δ,q_0,Z_0,F):
  - Q: states of the ε-NFA
  - ∑: input alphabet
  - Γ : stack symbols
  - δ: transition function
  - q_0: start state
  - Z_0: Initial stack top symbol
  - F: Final/accepting states

δ : The Transition Function

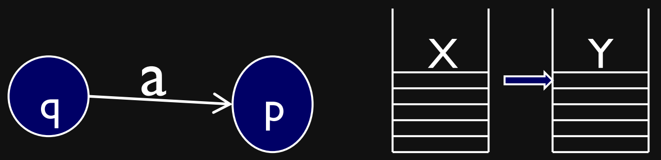
$\delta : Q \times \sum \cup \{\varepsilon\} \times \Gamma \cup \{\varepsilon\} \Rightarrow Q \times \Gamma$

old state *input symb.* *Stack top* *new state(s)* *new Stack top(s)*

δ(q,a,X) = {(p,Y), …}

Non-determinism

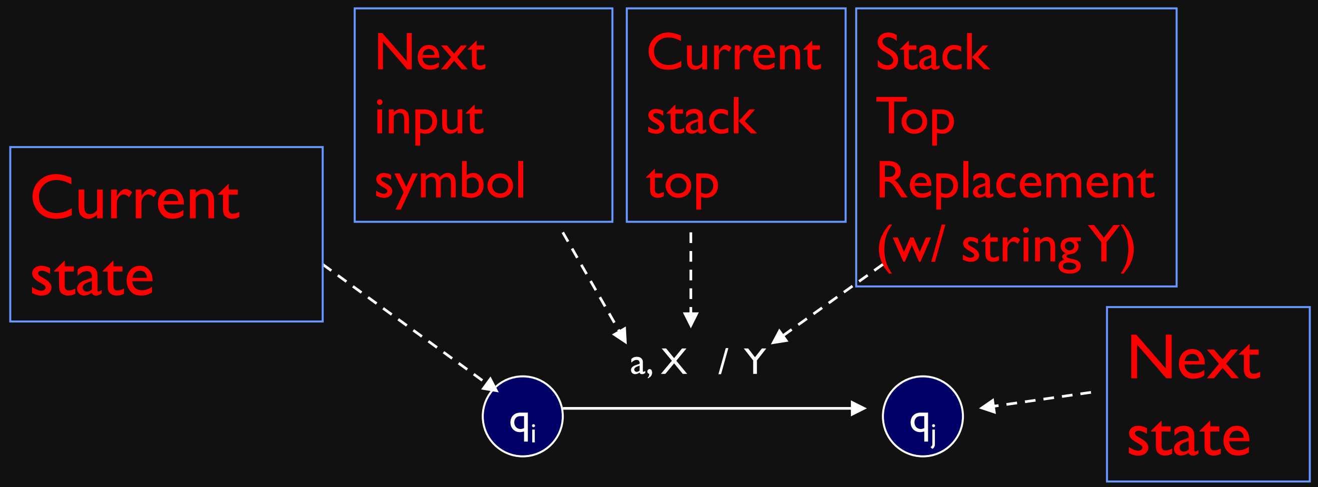
1. state transition from q to p
2. a is the next input symbol
3. X is the current stack *top* symbol
4. Y is the replacement for X; it is in Γ^* (a string of stack symbols)
   i. Set Y = ε for: Pop(X)
   ii. If Y=X: stack top is unchanged
   iii. If $Y=Z_1Z_2\ldots Z_k$: X is popped and is replaced by Y in reverse order (i.e., Z_1 will be the new stack top)



	Y = ?	**Action**
i)	Y=ε	Pop(X)
ii)	Y=X	Pop(X) Push(X)
iii)	$Y=Z_1Z_2..Z_k$	Pop(X) Push(Z_k) Push(Z_{k-1}) … Push(Z_2) Push(Z_1)

PDA as a state diagram

$\delta(q_i, a, X) = \{(q_j, Y)\}$



Example 1

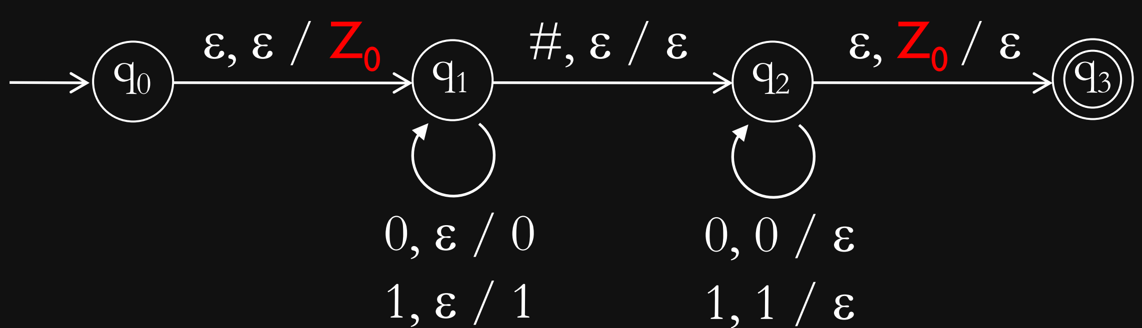
$$L = \{w\#w^R : w \in \{0, 1\}^*\}$$

$$\Sigma = \{0, 1, \#\}$$

$$\Gamma = \{0, 1\}$$

$\#,\ 0\#0,\ 01\#10 \in L$

$\varepsilon,\ 01\#1,\ 0\#\#0 \notin L$



write w on stack

read w^R from stack

Example-1'

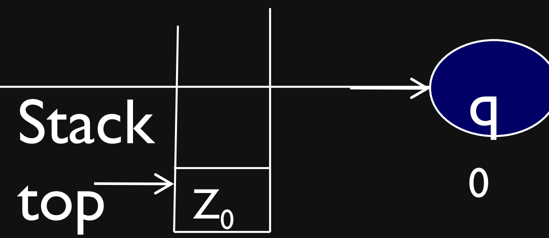
Let $L_{wwr} = \{ww^R \mid w \text{ is in } (0+1)^*\}$

- CFG for L_{wwr} : $S \Rightarrow 0S0 \mid 1S1 \mid \varepsilon$
- PDA for L_{wwr} :
- $P := (Q, \sum, \Gamma, \delta, q_0, Z_0, F)$

$= (\{q_0, q_1, q_2\}, \{0,1\}, \{0,1,Z_0\}, \delta, q_0, Z_0, \{q_2\})$

PDA for L_{wwr}

Initial state of the PDA:



1. $\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$
2. $\delta(q_0, 1, Z_0) = \{(q_0, 1Z_0)\}$

First symbol push on stack

3. $\delta(q_0, 0, 0) = \{(q_0, 00)\}$
4. $\delta(q_0, 0, 1) = \{(q_0, 01)\}$
5. $\delta(q_0, 1, 0) = \{(q_0, 10)\}$
6. $\delta(q_0, 1, 1) = \{(q_0, 11)\}$

Grow the stack by pushing new symbols on top of old (w-part)

7. $\delta(q_0, \varepsilon, 0) = \{(q_1, 0)\}$
8. $\delta(q_0, \varepsilon, 1) = \{(q_1, 1)\}$
9. $\delta(q_0, \varepsilon, Z_0) = \{(q_1, Z_0)\}$

Switch to popping mode, nondeterministically (boundary between w and w^R)

10. $\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\}$
11. $\delta(q_1, 1, 1) = \{(q_1, \varepsilon)\}$

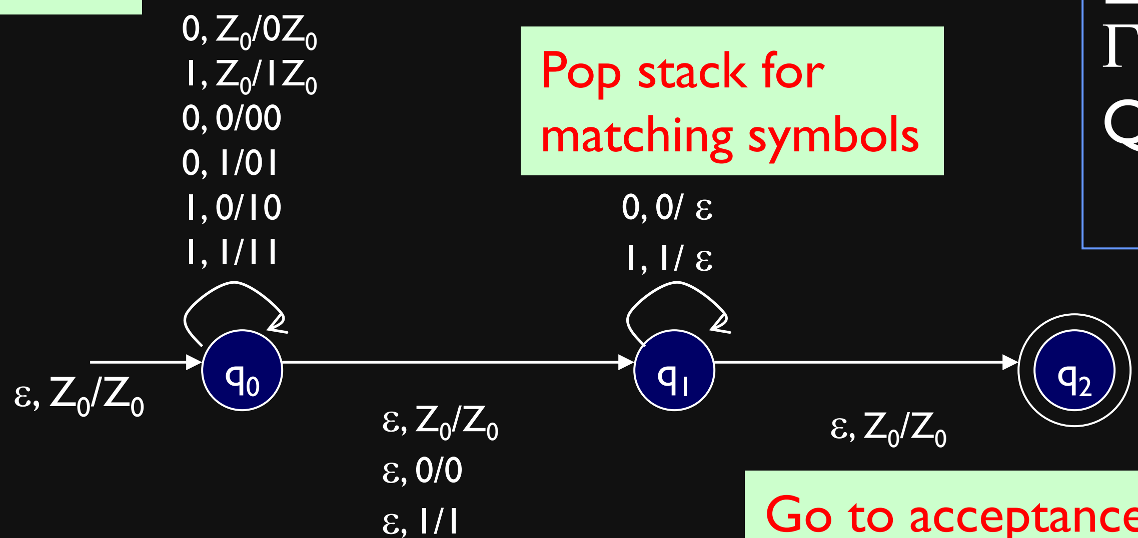
Shrink the stack by popping matching symbols (w^R-part)

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

Enter acceptance state

PDA for L_{wwr}: Transition Diagram

Grow stack



Pop stack for matching symbols

Switch to popping mode

Go to acceptance

$\sum = \{0, 1\}$

$\Gamma = \{Z_0, 0, 1\}$

$Q = \{q_0, q_1, q_2\}$

This would be a non-deterministic PDA

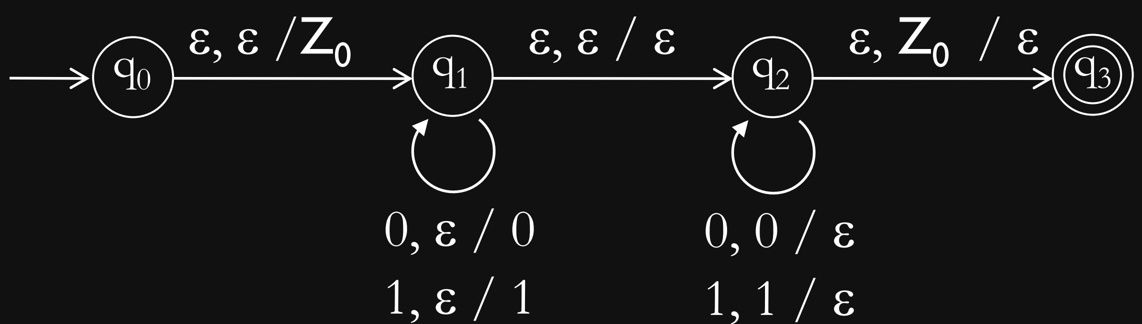
Another Design

$L = \{ww^R: w \in \Sigma^*\}$

$\Sigma = \{0, 1\}$

$\varepsilon, 00, 0110 \in L$

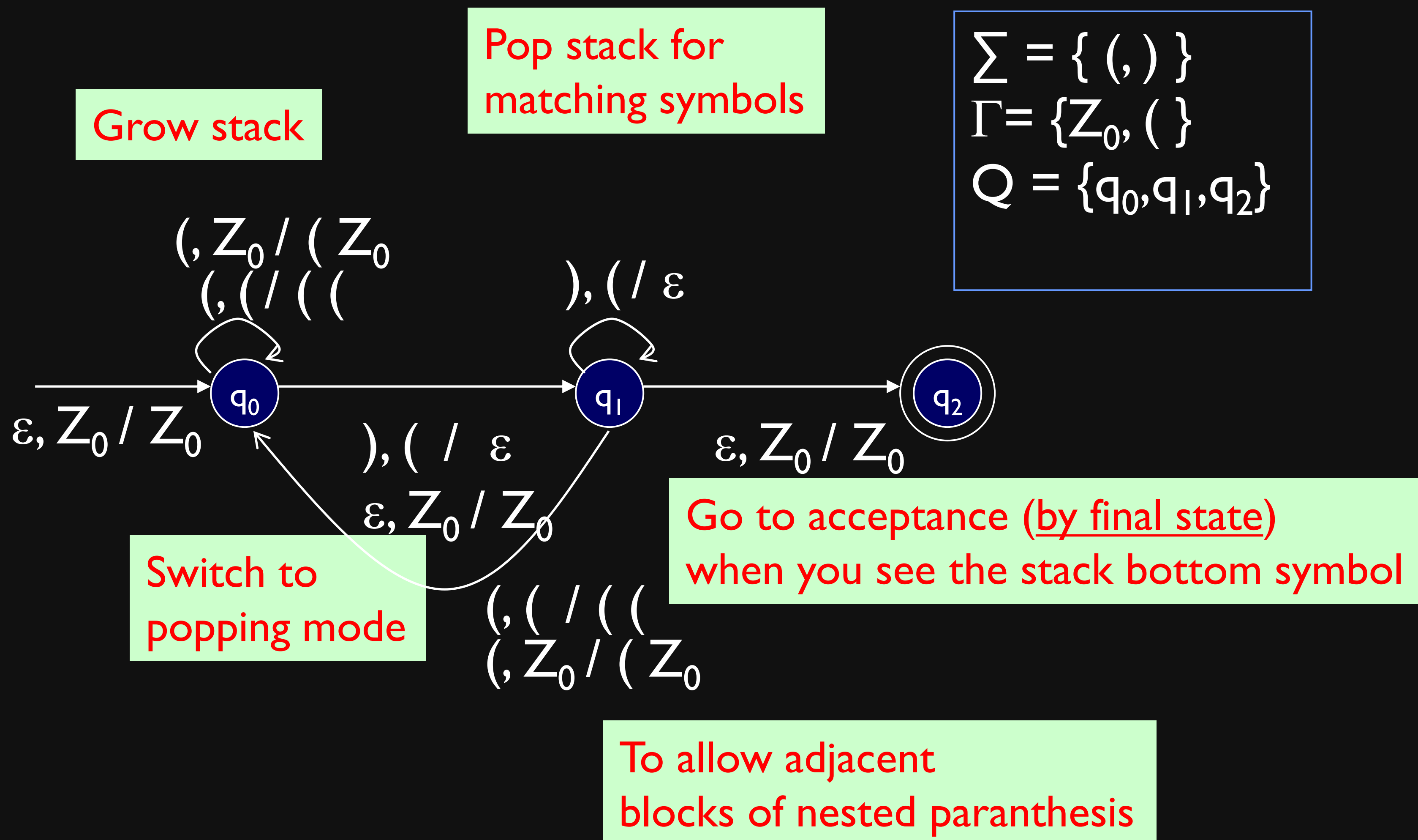
$1, 011, 010 \notin L$



guess middle of string

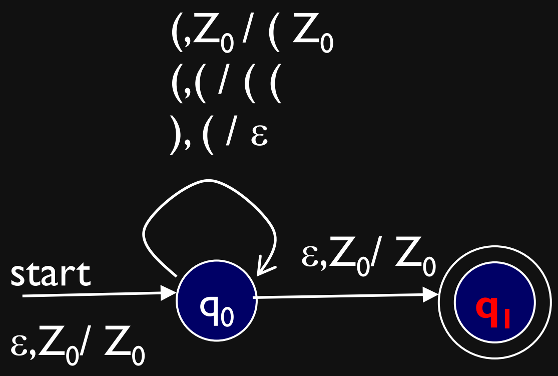
Example 2: language of balanced paranthesis

$\sum = \{ (,) \}$
$\Gamma = \{Z_0, (\}$
$Q = \{q_0, q_1, q_2\}$



I

Example 2: language of balanced parenthesis (another design)



$\sum = \{ (,) \}$

$\Gamma = \{Z_0, (\}$

$Q = \{q_0, q_1\}$

Example 3

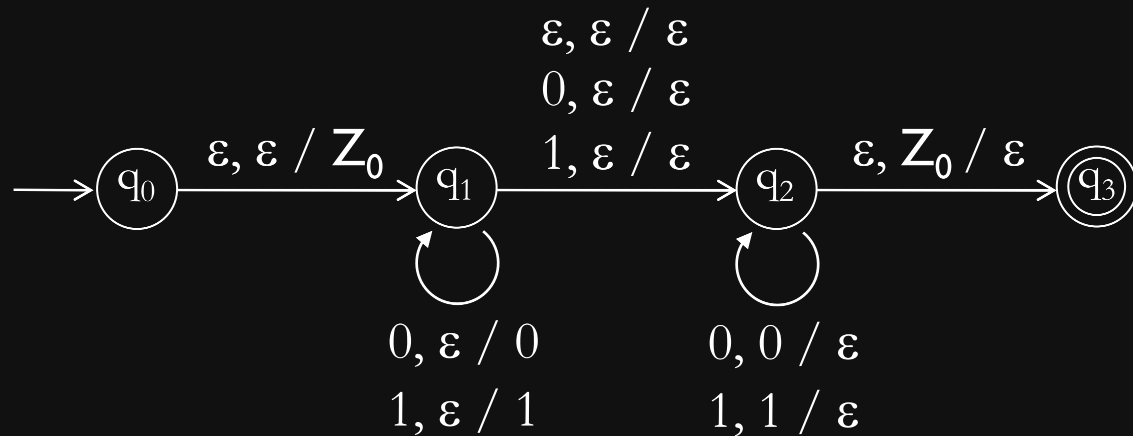
$$\boxed{L = \{w\!: w = w^R,\ w \in \Sigma^*\}}$$

$$\Sigma = \{0, 1\}$$

$\varepsilon, 1, 00, 010, 0110 \in L$

$011 \notin L$

$\underbrace{01101}_{x}\underbrace{10110}_{x^R}$ or $\underbrace{0110}_{x}1\underbrace{0110}_{x^R}$

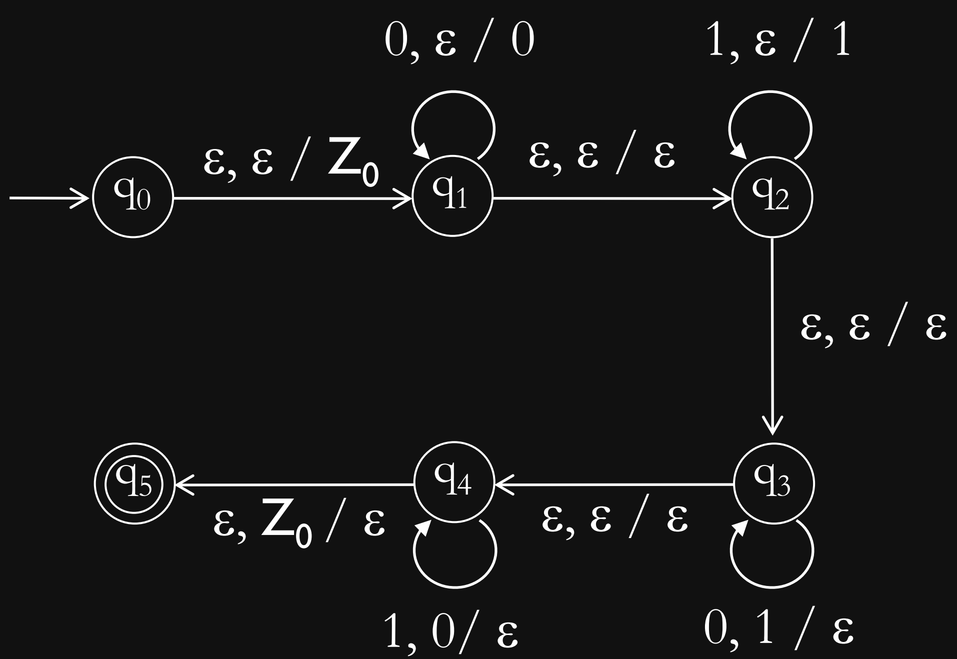


middle symbol can be ε, 0, or 1

Example 4

$$L = \{0^n 1^m 0^m 1^n \mid n \geq 0, m \geq 0\}$$

$$\Sigma = \{0, 1\}$$



input: $0^n\, 1^m\, 0^m\, 1^n$

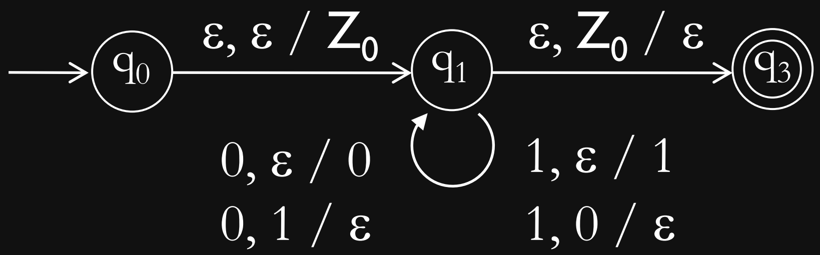
stack: $0^n\, 1^m$

Example 5

$L = \{w\text{: } w \text{ has same number 0s and 1s}\}$ $\Sigma = \{0, 1\}$

Strategy: Stack keeps track of excess of 0s or 1s

If at the end, stack is empty, number is equal

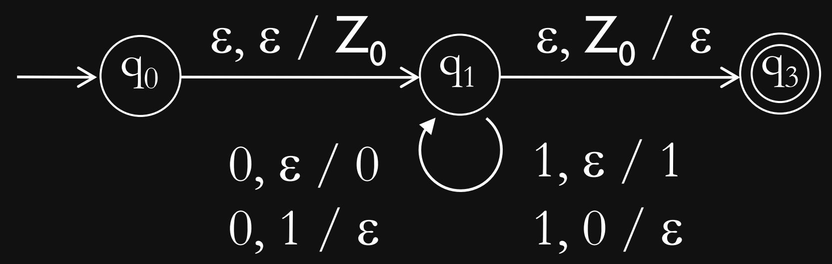


Example 5

$L = \{w\colon w$ has same number 0s and 1s$\}$ $\Sigma = \{0, 1\}$

Invariant: In every execution of the PDA:

#1 – #0 on stack = #1 – #0 in input so far



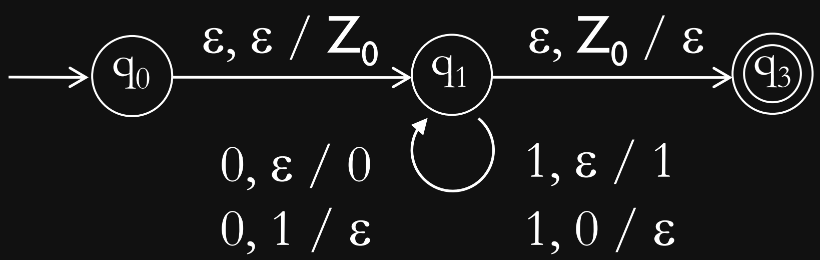
If w is not in L, it must be rejected

Example 5

$L = \{w\text{: } w \text{ has same number 0s and 1s}\}$ $\Sigma = \{0, 1\}$

Property: In some execution of the PDA:

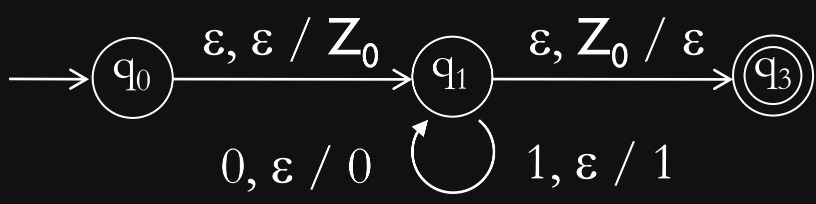
stack consists only of 0s or only of 1s (or ε)



If w is in L, some execution will accept

Example 5

$L = \{w\text{: } w \text{ has same number 0s and 1s}\}$ $\Sigma = \{0, 1\}$



$w = 001110$

read	stack
0	$Z_0\,0$
0	$Z_0\,00$
1	$Z_0\,0$
1	Z_0
1	$Z_0\,1$
0	Z_0

Example 6

$L = \{w$: w has two 0-blocks with same number of 0s

01011, 001011001, 10010101001 — allowed

01001000, 01111 — not allowed

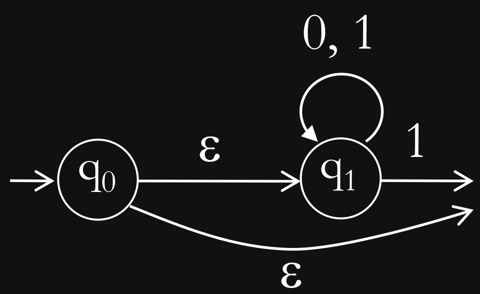
Strategy: Detect start of first 0-block

Push 0s on stack

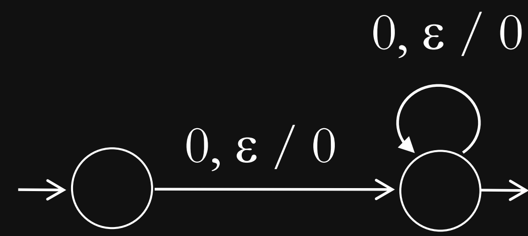
Detect start of second 0-block

Pop 0s from stack

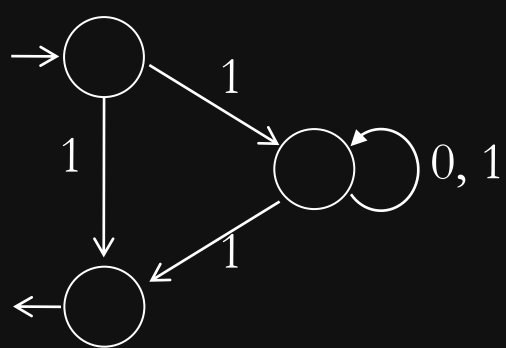
Example 6



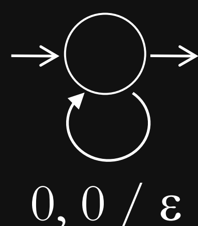
1 Detect start of first 0-block



2 Push 0s on stack



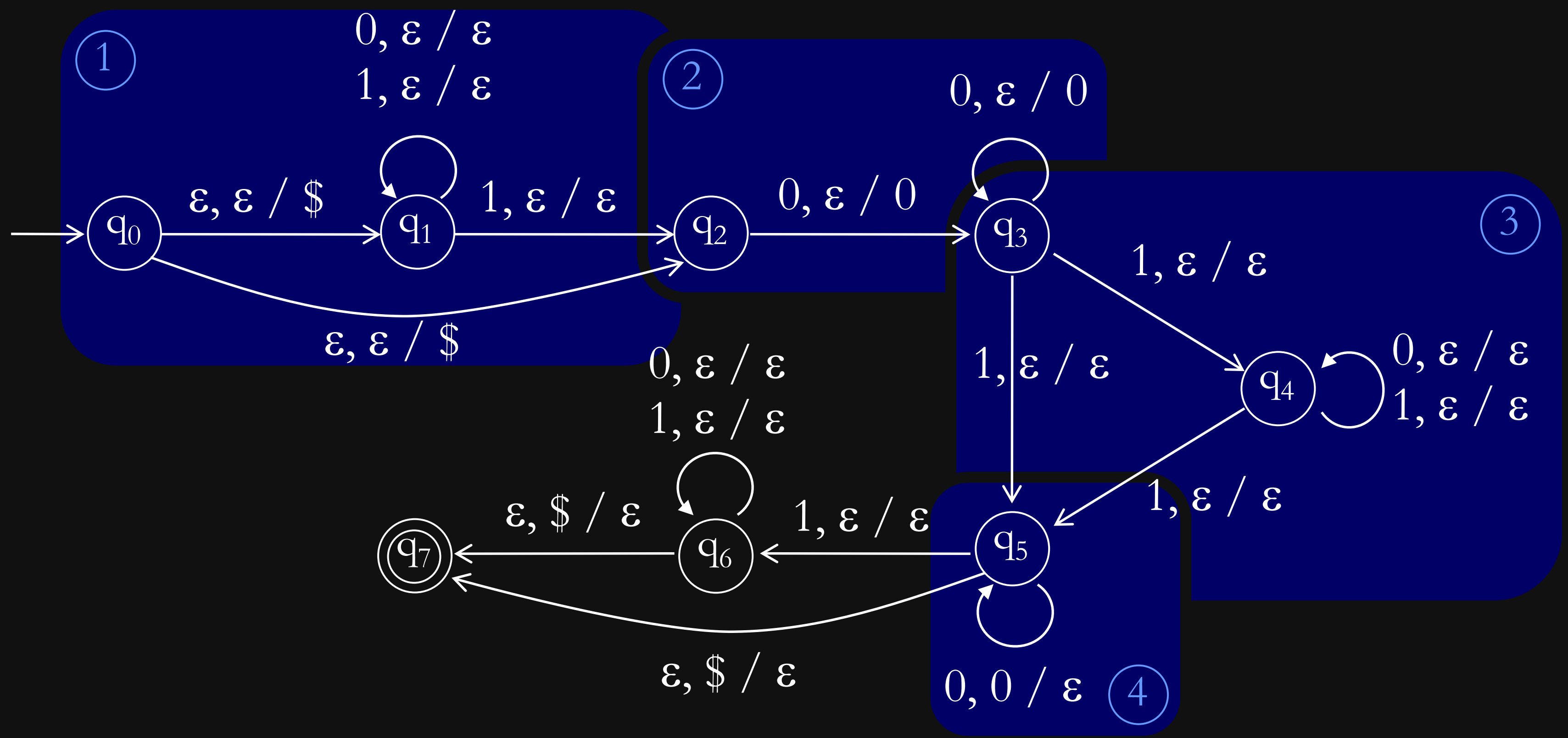
3 Detect start of second 0-block



4 Pop 0s from stack

Example 6

$L = \{w$: w has two 0-blocks with same number of 0s



CFG $\boxed{\leftrightarrow}$ PDA conversions

CFGs and PDAs

A language L is context-free if and only if it is accepted by some pushdown automaton.

context-free grammar $\rightleftarrows$ pushdown automaton

CFL Closure Properties

Closure Property Results

- CFLs are closed under:
  - Union
  - Concatenation
  - Kleene closure operator
  - reversal

- CFLs are *not* closed under:
  - Intersection
  - Difference
  - Complementation

Note: Reg languages are closed under these operators

CFLs are closed under ***union***

Let L_1 and L_2 be CFLs

<u>To show:</u> L_2 U L_2 is also a CFL

- Let S_1 and S_2 be the starting variables of the grammars for L_1 and L_2
  - Then, **S_{new} => S_1 | S_2**

CFLs are closed under ***concatenation***

- Let L_1 and L_2 be CFLs

for L_1 L_2,

S_{new}=> S1.S2

CFLs are closed under ***Kleene Closure***

- Let L be a CFL

  – Then for , L*,

  $s_{new} = s_{old} \cdot s_{new}$

CFLs are closed under *Reversal*

- Let L be a CFL, with grammar G=(V,T,P,S)
- For L^R, construct G^R=(V,T,P^R,S) s.t.,
  - If A==> α is in P, then:
    - A==> α^R is in P^R
    - (that is, reverse every production)