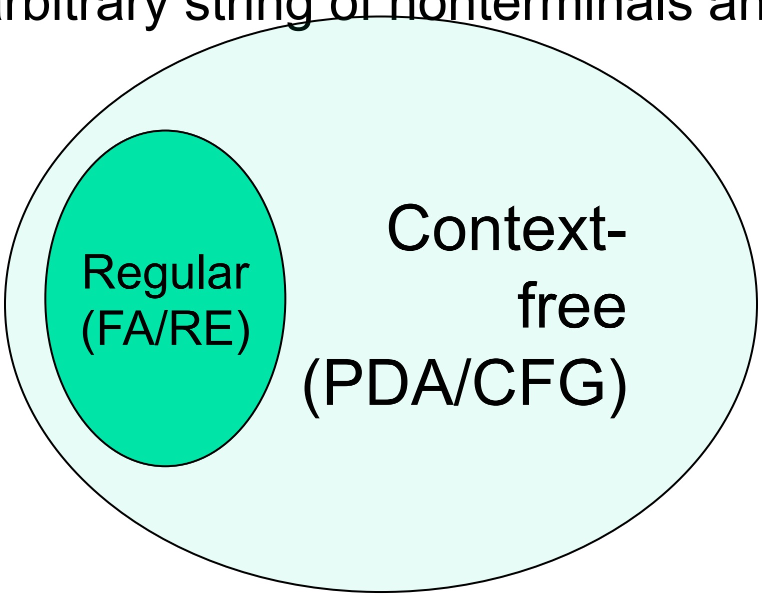
Pushdown Automata (PDA)

The structure and the content of the lecture is based on http://www.eecs.wsu.edu/~ananth/CptS317/Lectures/index.htm

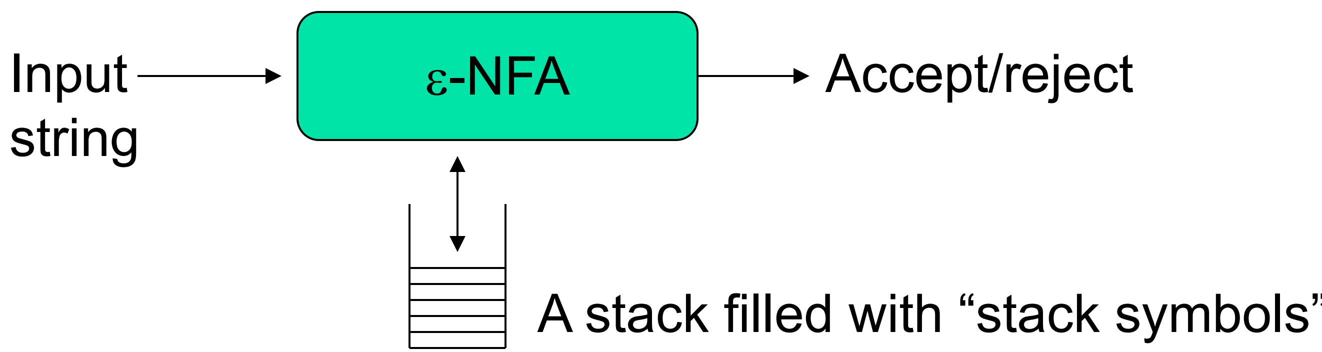
Excursion: Previous lecture

- Context-free grammar G=(V,T,P,S), where:
  - V: set of variables or non-terminals
  - T: set of terminals (= alphabet U {ε})
  - P: set of *productions,* each of which is of the form
    V ==> α_1 | α_2 | …
    - Where each α_i is an arbitrary string of nonterminals and terminals
  - S: start variable



PDA - the automata for CFLs

- What is?
  - What FA is to Reg Lang, PDA is to CFL
- PDA == [ε -NFA + "a stack"]
- Why a stack?



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

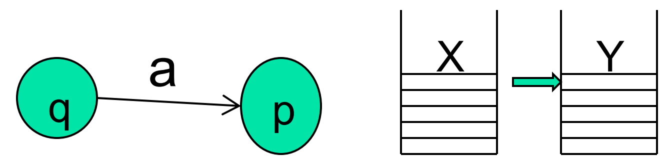
old state input symb. stack top new state(s) new Stack top(s)

$$\delta : Q \times \Sigma \times \Gamma \Rightarrow Q \times \Gamma$$

δ : The Transition Function

δ(q,a,X) = {(p,Y), …}

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)

Example (palindrome)

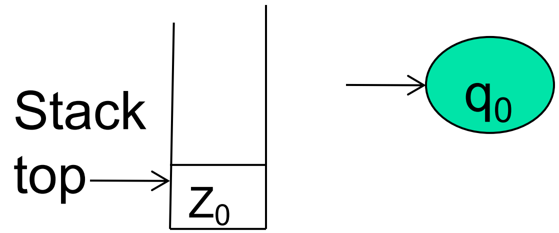
Let $L_{wwr} = \{ww^R \mid w \text{ is in } \{0,1\}^*\}$

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

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

Mark the botom of the stack

Initial state of the PDA:



PDA for L_{wwr}

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

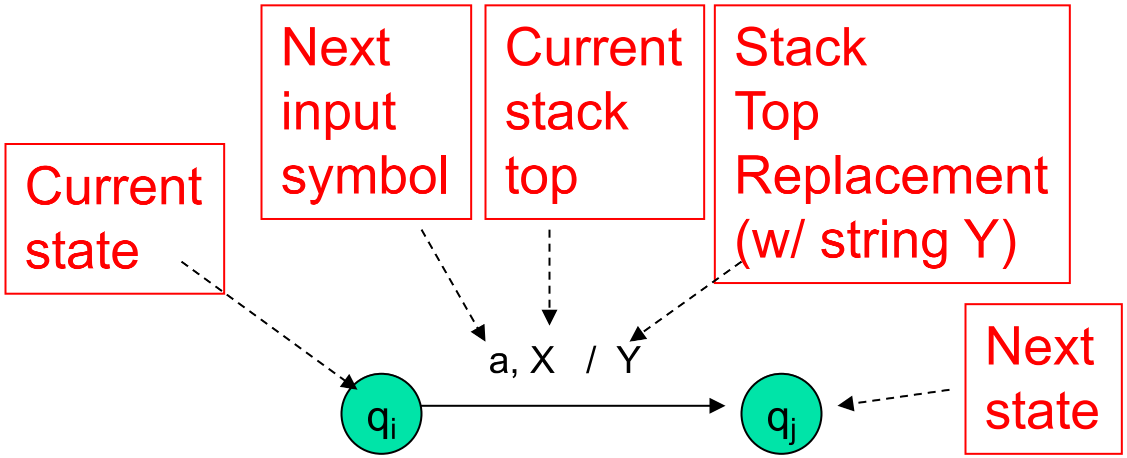
Shrink the stack by popping matching symbols (w^R-part)

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

Enter acceptance state

PDA as a state diagram

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

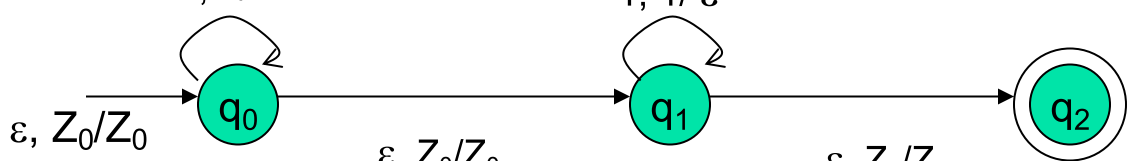


PDA for L_{wwr}: Transition Diagram

$\sum = \{0, 1\}$
$\Gamma = \{Z_0, 0, 1\}$
$Q = \{q_0, q_1, q_2\}$

Grow stack

Pop stack for matching symbols



Go to acceptance

Switch to popping mode

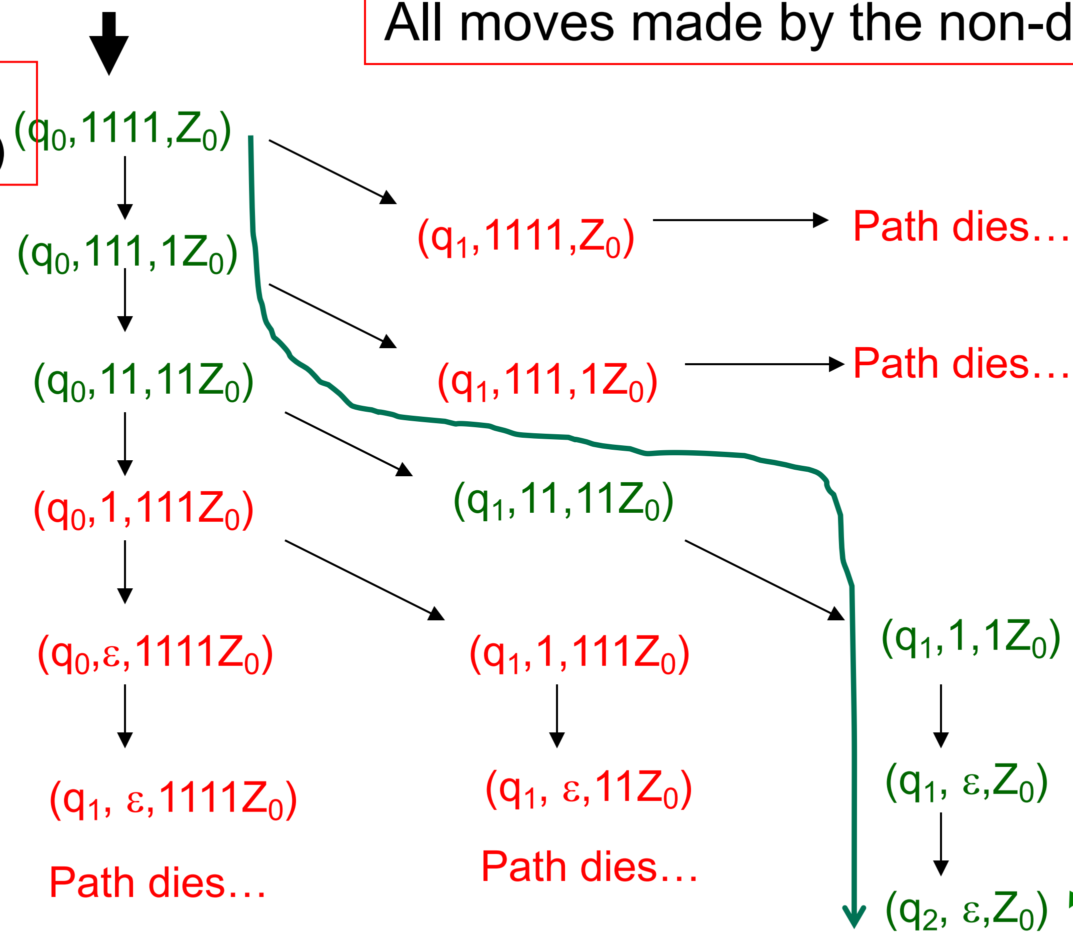
- Push input symbols onto the stack
- Non-deterministically move to a popping state (with or without consuming a single input symbol)
- If next input symbol is same as top of stack, pop
- If Z_0 on top of stack move to accept state

Non-deterministic PDA: 2 output transitions i.e.:
$(q_0, 0, 0) = (q_0, 0), (q_0, \varepsilon, 0) = (q_1, 0)$:

How does the PDA for L_{wwr} work on input “1111”?

All moves made by the non-deterministic PDA

Instantaneous Description (ID)



__Acceptance by final state:__

= *empty input*
AND
final state

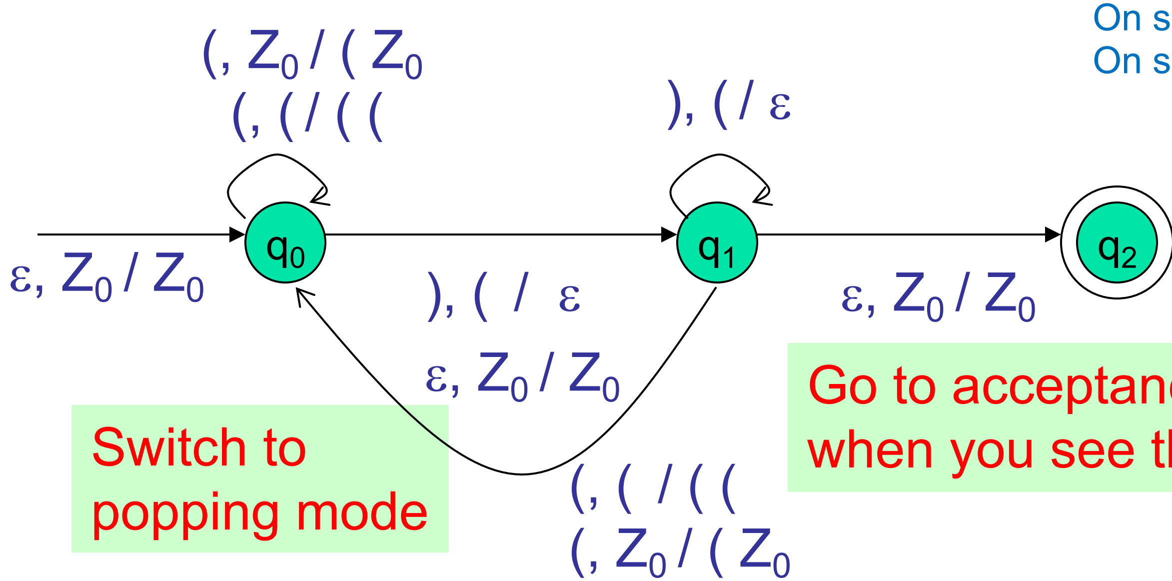
Example 2: language of balanced paranthesis

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

On seeing a (push it onto the stack
On seeing a) pop if a (is in the stack

Grow stack

Pop stack for matching symbols

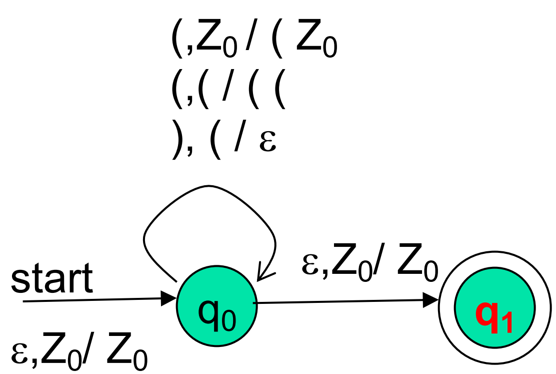


Switch to popping mode

Go to acceptance (by final state) when you see the stack bottom symbo

To allow adjacent blocks of nested paranthesis

Example 2: language of balanced paranthesis (another design)



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

There are two types of PDAs that one can design:
those that accept by final state or by empty stack

Acceptance by…

- ***PDAs that accept by final state:***
  - For a PDA P, the language accepted by P, denoted by L(P) by *final state*, is:
    - $\{w \mid (q_0,w,Z_0) \mid\text{---}^* (q,\varepsilon, A) \}$, s.t., $q \in F$

Checklist:
- input exhausted?
- in a final state?

- ***PDAs that accept by empty stack:***
  - For a PDA P, the language accepted by P, denoted by N(P) by *empty stack*, is:
    - $\{w \mid (q_0,w,Z_0) \mid\text{---}^* (q, \varepsilon, \varepsilon) \}$, for any $q \in Q$.

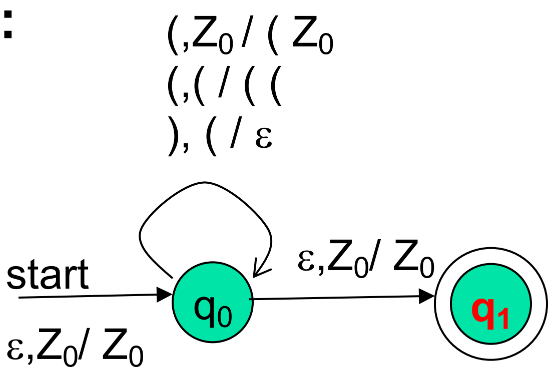
Q) Does a PDA that accepts by empty stack need any final state specified in the design?

Checklist:
- input exhausted?
- is the stack empty?

Example: L of balanced parenthesis

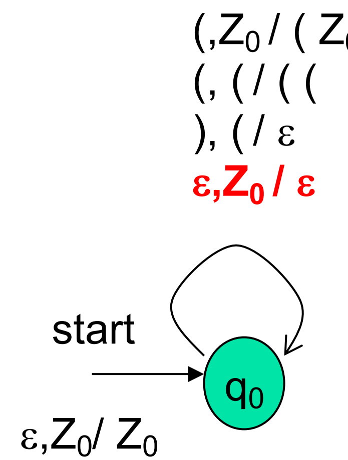
<u>PDA that accepts by final state</u>

P_F:



<u>An equivalent PDA that accepts by empty stack</u>

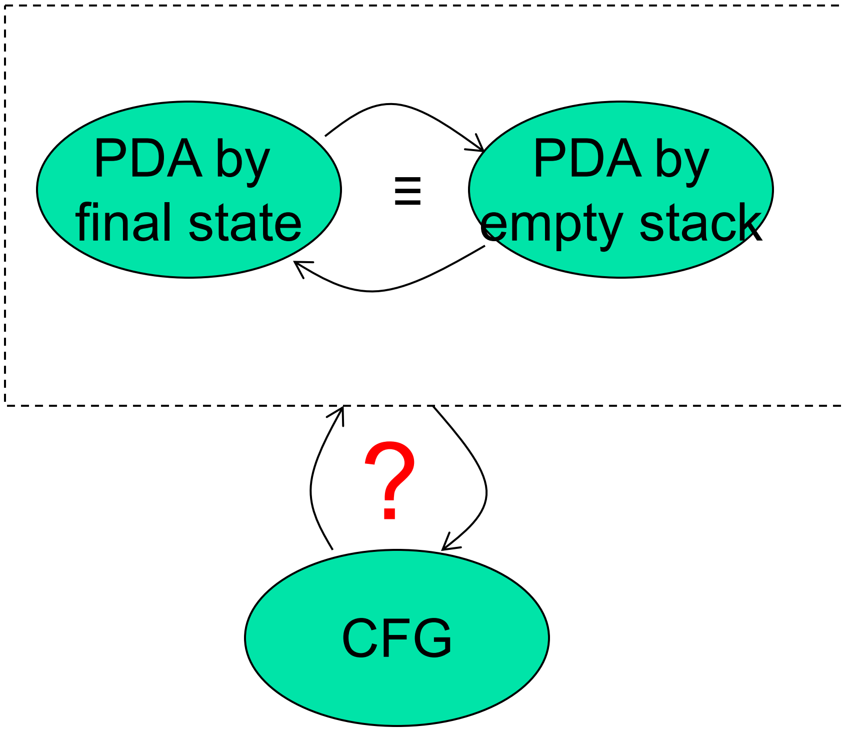
P_N:



How will these two PDAs work on the input: ((( )) ( )) ( )

Equivalence of PDAs and CFGs

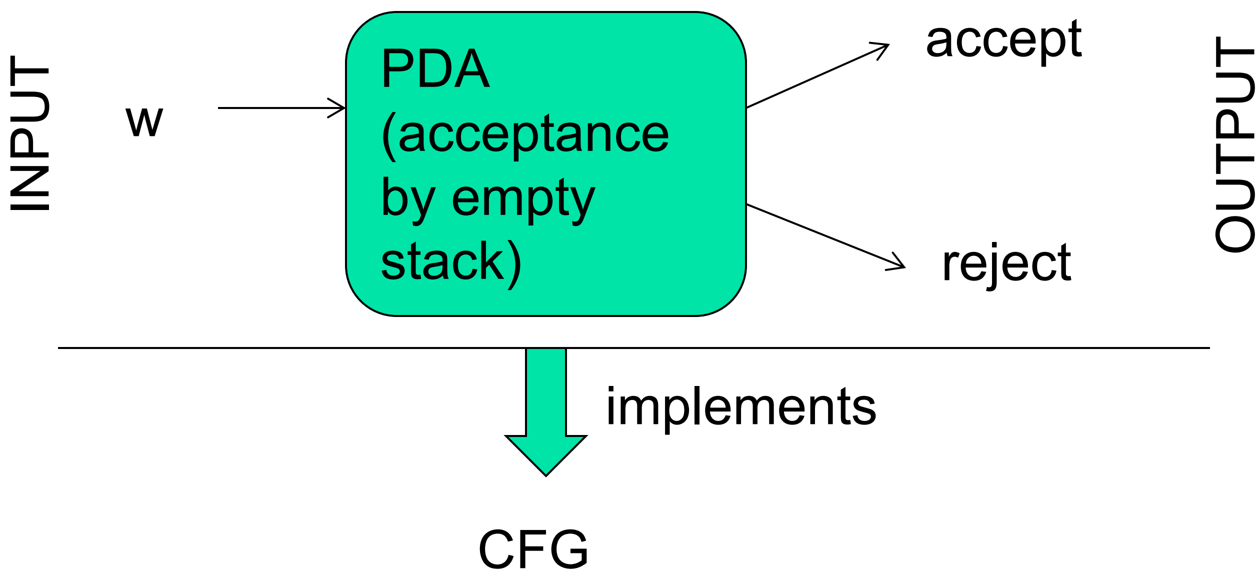
CFGs == PDAs ==> CFLs



This is same as: "implementing a CFG using a PDA"

Converting CFG to PDA

Main idea: The PDA simulates the leftmost derivation on a given w, and upon consuming it fully it either arrives at acceptance (by empty stack) or non-acceptance.



This is same as: "implementing a CFG using a PDA"

Converting a CFG into a PDA

<u>Main idea:</u> The PDA simulates the leftmost derivation on a given w, and upon consuming it fully it either arrives at acceptance (by <u>empty stack</u>) or non-acceptance.

<u>Steps:</u>

1. Push the right hand side of the production onto the stack, with leftmost symbol at the stack top
2. If stack top is the leftmost variable, then replace it by all its productions (each possible substitution will represent a *distinct* path taken by the non-deterministic PDA)
3. If stack top has a terminal symbol, and if it matches with the next symbol in the input string, then pop it.

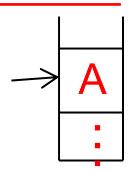
PDA by final state: (Q, ∑, Γ, δ, q_0, Z_0, F)

Formal construction of PDA from CFG

Note: Initial stack symbol (S) same as the start variable in the grammar

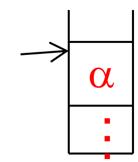
- Given: G= (V_N,V_T, S, P)
- Output: P_N = ({q}, V_T, V_N U V_T, δ, q, S)
- δ:

Before:

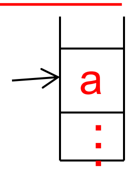


- For all A ∈ V_N , add the following transition(s) in the PDA:
  - δ(q, ε ,A) = { (q, α) | "A -->α" ∈ P}

After:

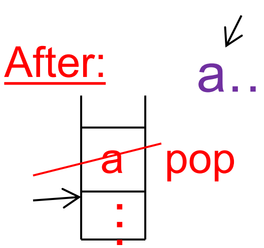


Before:



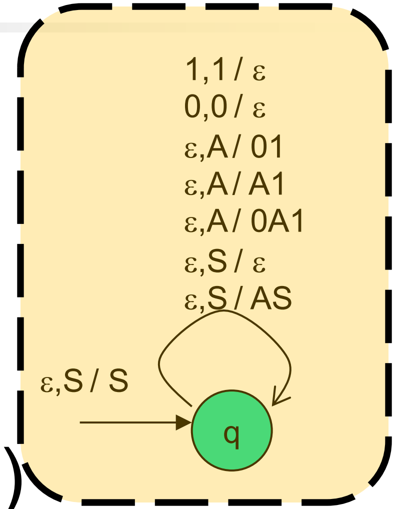
- For all *a* ∈ V_T, add the following transition(s) in the PDA:
  - δ(q,*a*,*a*)= { (q, ε) }

After:



Example: CFG to PDA

- G = ({S,A}, {0,1}, P, S)
- P:
  - S --> AS | ε
  - A --> 0A1 | A1 | 01
- PDA = ({q}, {0,1}, {0,1,A,S}, δ, q, S)
- δ:
  - δ(q, ε , S) = { (q, AS), (q, ε)}
  - δ(q, ε , A) = { (q,0A1), (q,A1), (q,01) }
  - δ(q, 0, 0) = { (q, ε) }
  - δ(q, 1, 1) = { (q, ε) }



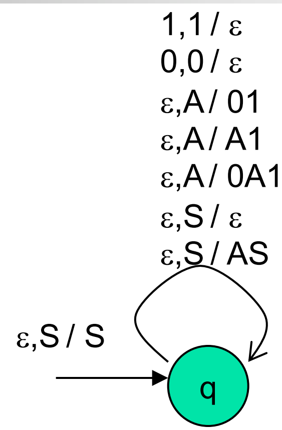
How will this new PDA work?

Lets simulate string <u>0011</u>

Simulating string 0011 on the new PDA …

PDA (δ):

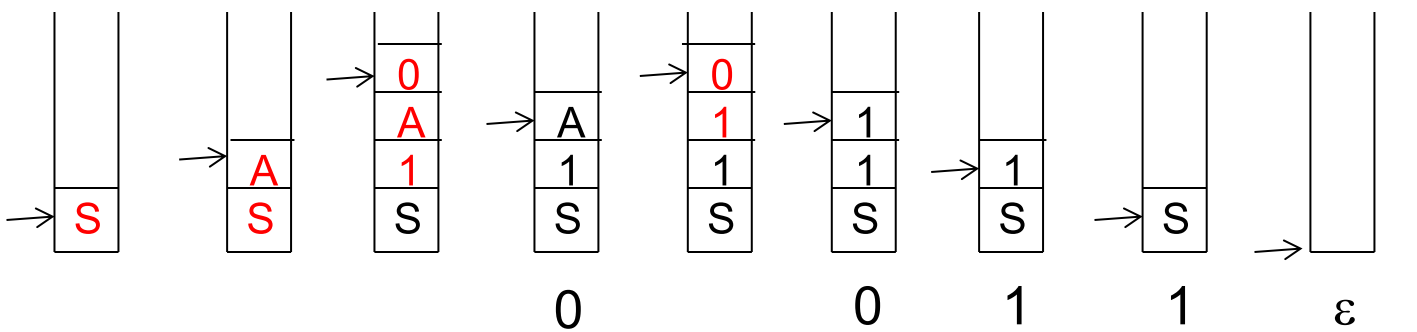
δ(q, ε , S) = { (q, AS), (q, ε)}
δ(q, ε , A) = { (q,0A1), (q,A1), (q,01) }
δ(q, 0, 0) = { (q, ε) }
δ(q, 1, 1) = { (q, ε) }



Leftmost deriv.:

S => AS
=> 0A1S
=> 0011S
=> 0011

Stack moves (shows only the successful path):



Accept by empty stack

S =>AS =>0A1S =>0011S => 0011

Summary

- PDA
  - Definition
  - With acceptance – by final state
  - With acceptance – by empty stack
- PDA (by final state) = PDA (by empty stack) <== CFG