

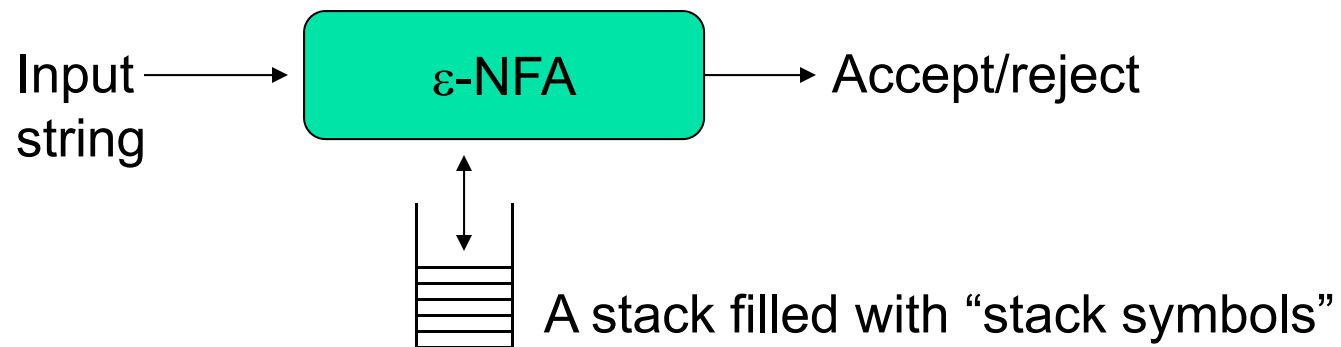


Pushdown Automata (PDA)

Reading: Chapter 6

PDA - the automata for CFLs

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





Pushdown Automata - Definition

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

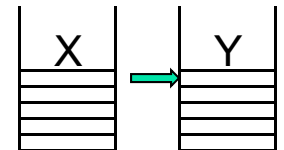
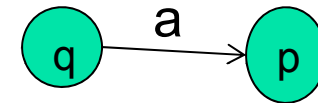
$$\delta : \overset{\text{old state}}{Q} \times \overset{\text{input symb.}}{\Sigma} \times \overset{\text{Stack top}}{\Gamma} \Rightarrow \overset{\text{new state(s)}}{Q} \times \overset{\text{new Stack top(s)}}{\Gamma}$$

δ : The Transition Function

$$\delta(q, a, X) = \{(p, Y), \dots\}$$

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)

Non-determinism



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



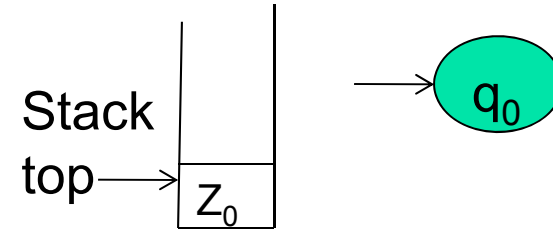
Example

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

- CFG for L_{ww^R} : $S \Rightarrow 0S0 \mid 1S1 \mid \varepsilon$
- PDA for L_{ww^R} :
- $P := (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$
 $= (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\})$

PDA for L_{ww^R}

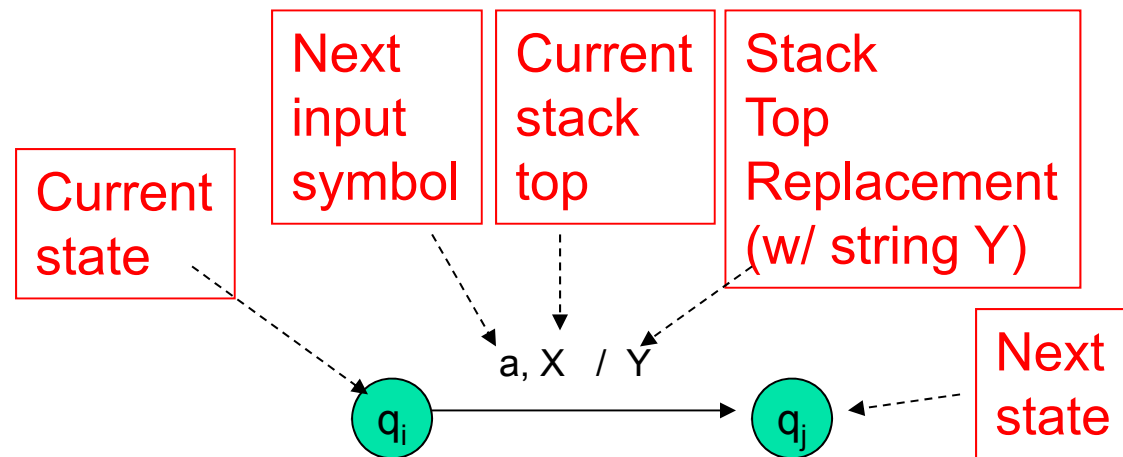
Initial state of the PDA:



- | | | | |
|-----|--|---|---|
| 1. | $\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$ | } | First symbol push on stack |
| 2. | $\delta(q_0, 1, Z_0) = \{(q_0, 1Z_0)\}$ | | |
| 3. | $\delta(q_0, 0, 0) = \{(q_0, 00)\}$ | } | Grow the stack by pushing
new symbols on top of old
(w-part) |
| 4. | $\delta(q_0, 0, 1) = \{(q_0, 01)\}$ | | |
| 5. | $\delta(q_0, 1, 0) = \{(q_0, 10)\}$ | | |
| 6. | $\delta(q_0, 1, 1) = \{(q_0, 11)\}$ | | |
| 7. | $\delta(q_0, \varepsilon, 0) = \{(q_1, 0)\}$ | } | Switch to popping mode, nondeterministically
(boundary between w and w^R) |
| 8. | $\delta(q_0, \varepsilon, 1) = \{(q_1, 1)\}$ | | |
| 9. | $\delta(q_0, \varepsilon, Z_0) = \{(q_1, Z_0)\}$ | | |
| 10. | $\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\}$ | } | Shrink the stack by popping matching
symbols (w^R -part) |
| 11. | $\delta(q_1, 1, 1) = \{(q_1, \varepsilon)\}$ | | |
| 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

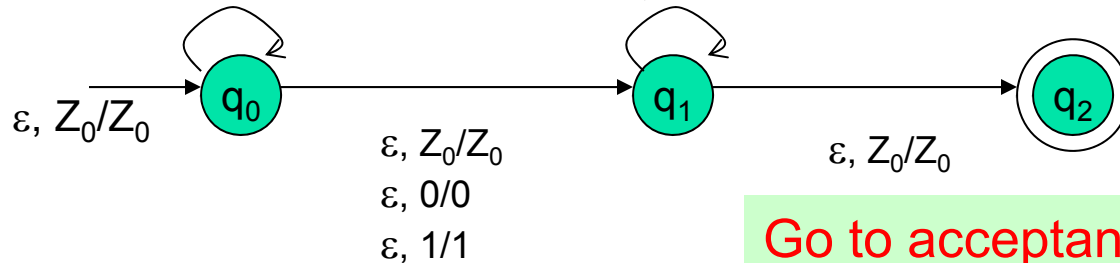
Grow stack

$0, Z_0/0Z_0$
 $1, Z_0/1Z_0$
 $0, 0/00$
 $0, 1/01$
 $1, 0/10$
 $1, 1/11$

Pop stack for
matching symbols

$0, 0/\epsilon$
 $1, 1/\epsilon$

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



Switch to
popping mode

Go to acceptance

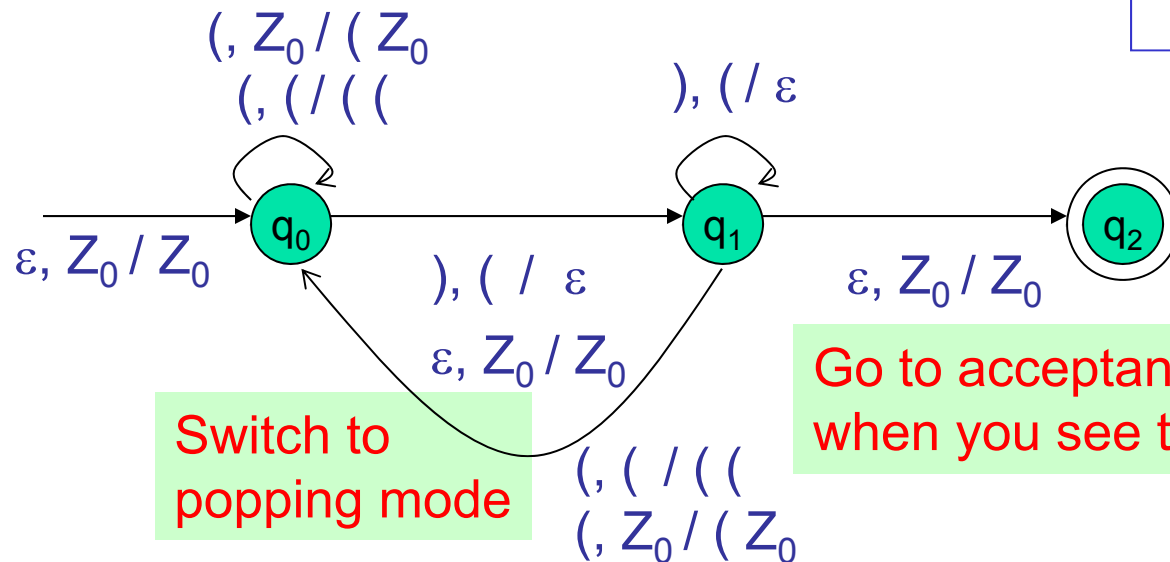
This would be a non-deterministic PDA

Example 2: language of balanced paranthesis

Grow stack

Pop stack for matching symbols

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

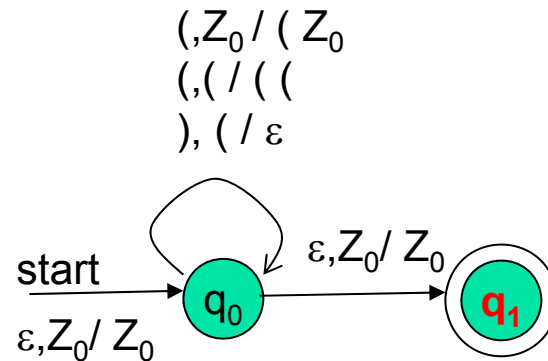


Switch to popping mode

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

To allow adjacent blocks of nested paranthesis

Example 2: language of balanced paranthesis (another design)



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



PDA's Instantaneous Description (ID)

A PDA has a configuration at any given instance:
(q,w,y)

- q - current state
 - w - remainder of the input (i.e., unconsumed part)
 - y - current stack contents as a string from top to bottom of stack
-

If $\delta(q, a, X) = \{(p, A)\}$ is a transition, then the following are also true:

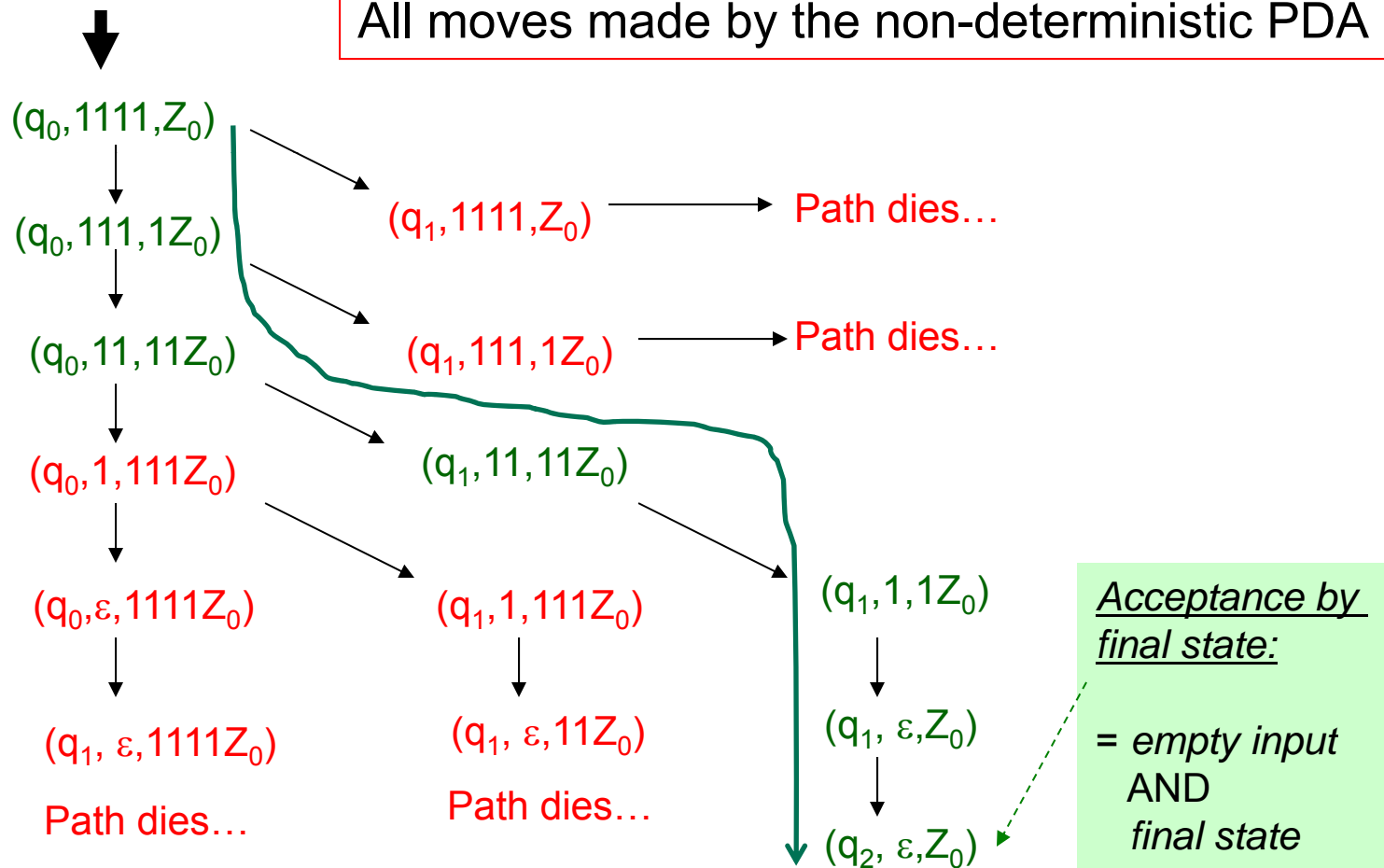
- $(q, a, X) \vdash (p, \varepsilon, A)$
 - $(q, aw, XB) \vdash (p, w, AB)$
-

\vdash sign is called a “turnstile notation” and represents one move

\vdash^* sign represents a sequence of moves

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

All moves made by the non-deterministic PDA





Principles about IDs

- Theorem 1: If for a PDA,
 $(q, x, A) \vdash^{*} (p, y, B)$, then for any string $w \in \Sigma^{*}$ and $\gamma \in \Gamma^{*}$, it is also true that:
 - $(q, x w, A \gamma) \vdash^{*} (p, y w, B \gamma)$
- Theorem 2: If for a PDA,
 $(q, x w, A) \vdash^{*} (p, y w, B)$, then it is also true that:
 - $(q, x, A) \vdash^{*} (p, y, B)$

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:

- $\{w \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, A)\}, \text{ s.t., } q \in F$

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:

- $\{w \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)\}, \text{ for any } q \in Q.$

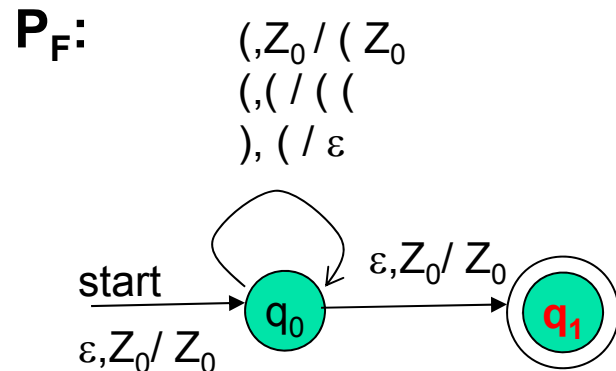
Q) Does a PDA that accepts by empty stack need any final state specified in the design?

Checklist:

- input exhausted?
- is the stack empty?

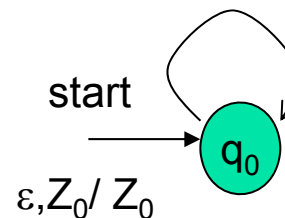
Example: L of balanced parenthesis

PDA that accepts by final state



An equivalent PDA that accepts by empty stack

P_N:

$$\begin{array}{l} (, Z_0 / (Z_0 \\ (, (/ ((\\), (/ \varepsilon \\ \varepsilon, Z_0 / \varepsilon \end{array}$$


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



PDA for L_{ww^R} : Proof of correctness

- Theorem: The PDA for L_{ww^R} accepts a string x by final state **if and only if** x is of the form ww^R .
- Proof:
 - *(if-part)* If the string is of the form ww^R then there exists a sequence of IDs that leads to a final state:
 $(q_0, ww^R, Z_0) \vdash^* (q_0, w^R, wZ_0) \vdash^* (q_1, w^R, wZ_0) \vdash^* (q_1, \varepsilon, Z_0) \vdash^* (\mathbf{q_2}, \varepsilon, Z_0)$
 - *(only-if part)*
 - Proof by induction on $|x|$



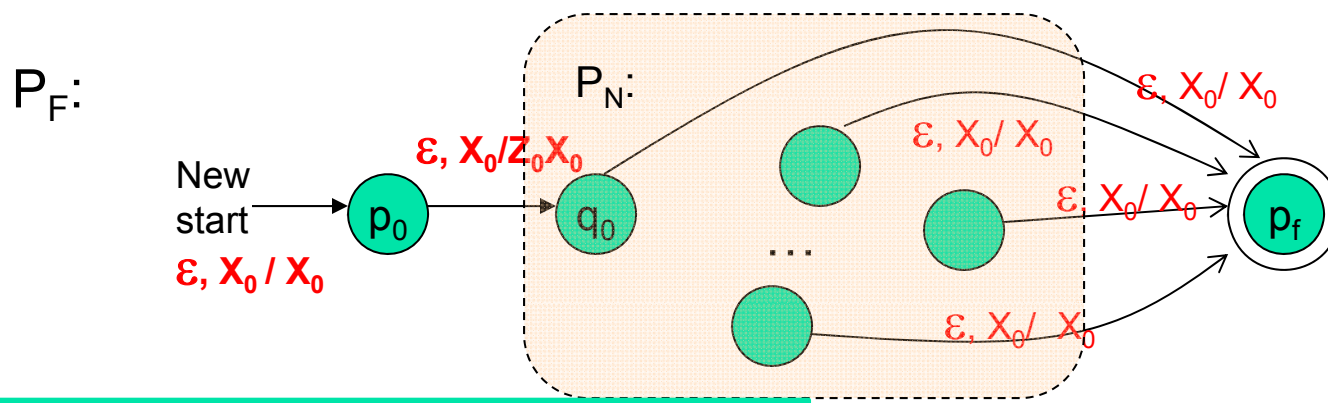
PDAs accepting by final state and empty stack are equivalent

- $P_F \leq$ PDA accepting by final state
 - $P_F = (Q_F, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$
- $P_N \leq$ PDA accepting by empty stack
 - $P_N = (Q_N, \Sigma, \Gamma, \delta_N, q_0, Z_0)$
- Theorem:
 - $(P_N \Rightarrow P_F)$ For every P_N , there exists a P_F s.t. $L(P_F) = L(P_N)$
 - $(P_F \Rightarrow P_N)$ For every P_F , there exists a P_N s.t. $L(P_F) = L(P_N)$

How to convert an empty stack PDA into a final state PDA?

$P_N \Rightarrow P_F$ construction

- Whenever P_N 's stack becomes empty, make P_F go to a final state without consuming any addition symbol
- To detect empty stack in P_N : P_F pushes a new stack symbol X_0 (not in Γ of P_N) initially before simulating P_N

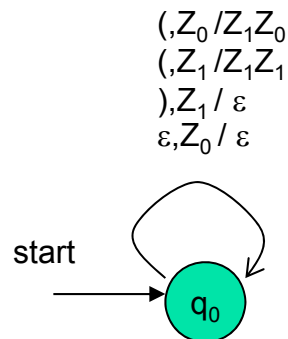


$$P_F = (Q_N \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$$

Example: Matching parenthesis “(” “)”

$P_N:$ ($\{q_0\}$, $\{(\cdot)\}$, $\{Z_0, Z_1\}$, δ_N , q_0 , Z_0)

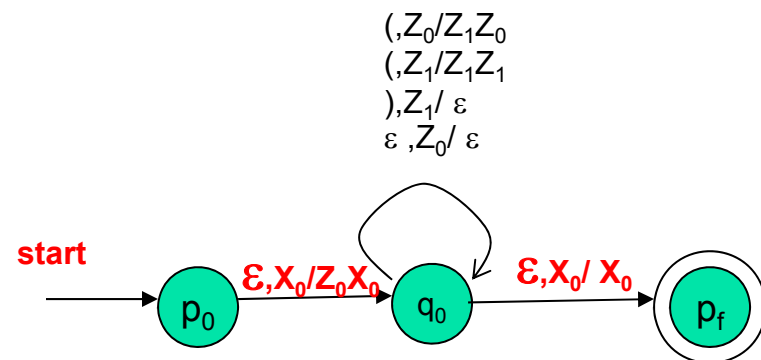
$\delta_N:$ $\delta_N(q_0, (\cdot, Z_0) = \{ (q_0, Z_1 Z_0) \}$
 $\delta_N(q_0, (\cdot, Z_1) = \{ (q_0, Z_1 Z_1) \}$
 $\delta_N(q_0, \cdot, Z_1) = \{ (q_0, \epsilon) \}$
 $\delta_N(q_0, \epsilon, Z_0) = \{ (q_0, \epsilon) \}$



Accept by empty stack

$P_f:$ ($\{p_0, q_0, p_f\}$, $\{(\cdot)\}$, $\{X_0, Z_0, Z_1\}$, δ_f , p_0 , X_0 , p_f)

$\delta_f:$ $\delta_f(p_0, \epsilon, X_0) = \{ (q_0, Z_0) \}$
 $\delta_f(q_0, (\cdot, Z_0) = \{ (q_0, Z_1 Z_0) \}$
 $\delta_f(q_0, (\cdot, Z_1) = \{ (q_0, Z_1 Z_1) \}$
 $\delta_f(q_0, \cdot, Z_1) = \{ (q_0, \epsilon) \}$
 $\delta_f(q_0, \epsilon, Z_0) = \{ (q_0, \epsilon) \}$
 $\delta_f(p_0, \epsilon, X_0) = \{ (p_f, X_0) \}$



Accept by final state

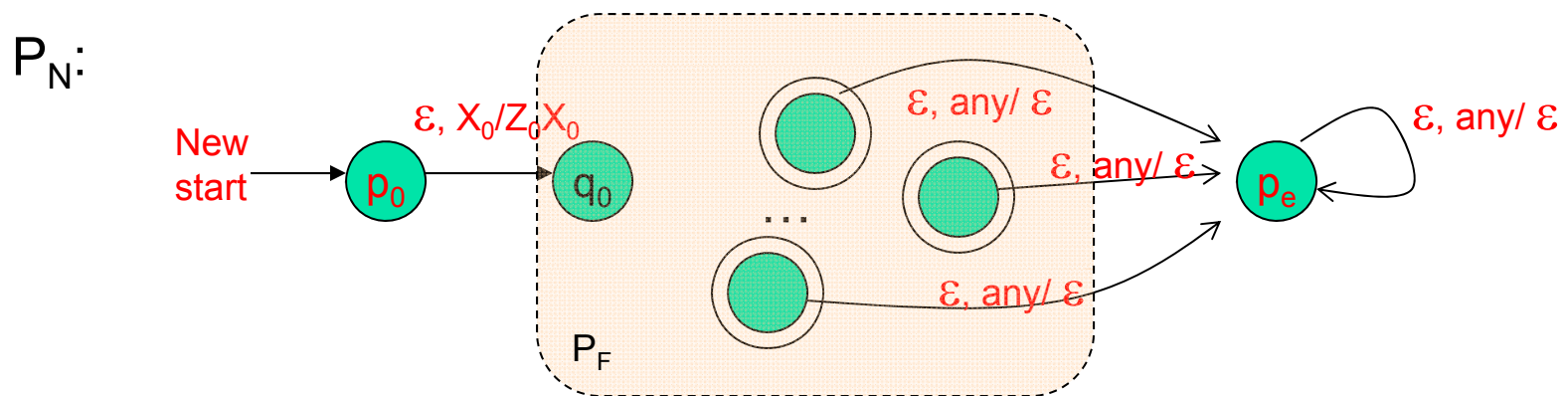
How to convert an final state PDA into an empty stack PDA?

$P_F \Rightarrow P_N$ construction

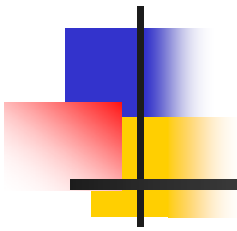
- Main idea:

- Whenever P_F reaches a final state, just make an ε -transition into a new end state, clear out the stack and accept
- Danger: What if P_F design is such that it clears the stack midway *without* entering a final state?
 → to address this, add a new start symbol X_0 (not in Γ of P_F)

$$P_N = (Q \cup \{p_0, p_e\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$$

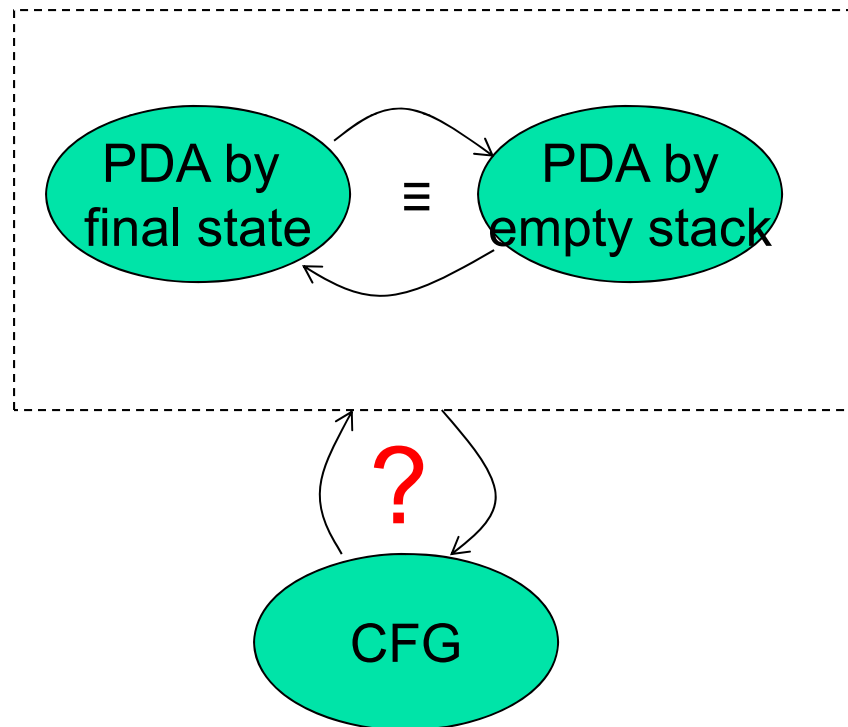


Equivalence of PDAs and CFGs





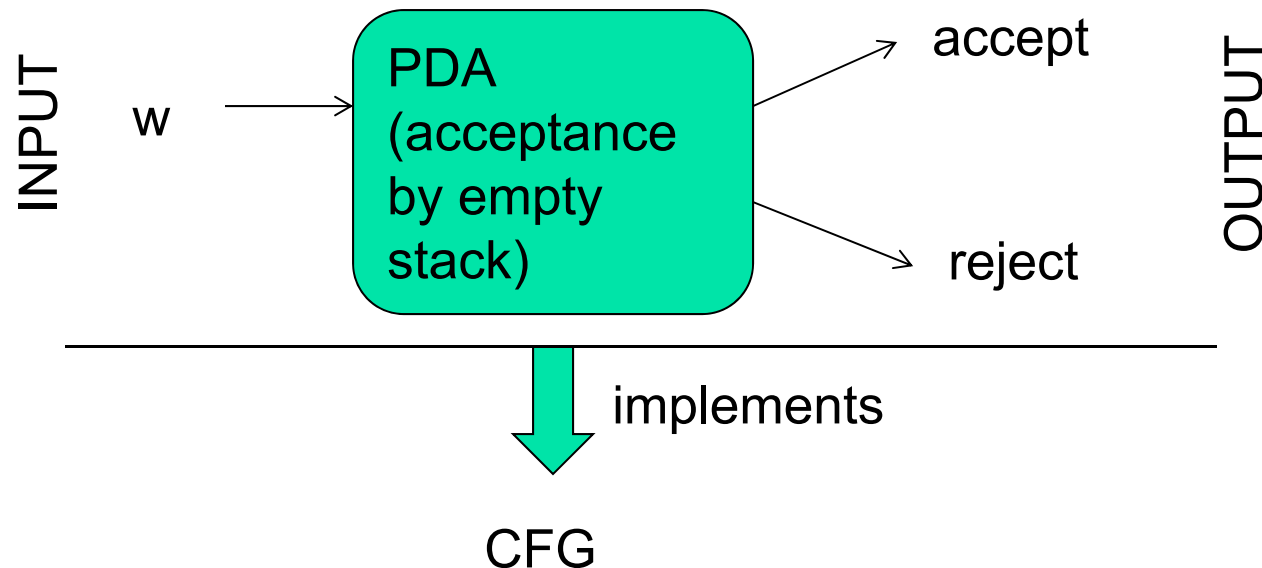
CFGs == PDAs ==> CFLs



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.



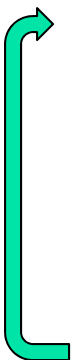
This is same as: “implementing a CFG using a PDA”



Converting a CFG into a PDA

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.

Steps:

- 
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

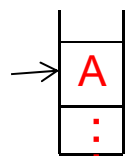
State is inconsequential (only one state is needed)

Formal construction of PDA from CFG

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

- Given: $G = (V, T, P, S)$
- Output: $P_N = (\{q\}, T, V \cup T, \delta, q, S)$
- δ :

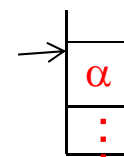
Before:



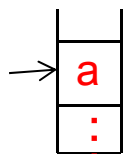
- For all $A \in V$, add the following transition(s) in the PDA:

- $\delta(q, \varepsilon, A) = \{ (q, \alpha) \mid "A \Rightarrow \alpha" \in P \}$

After:



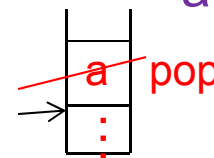
Before:



- For all $a \in T$, add the following transition(s) in the PDA:

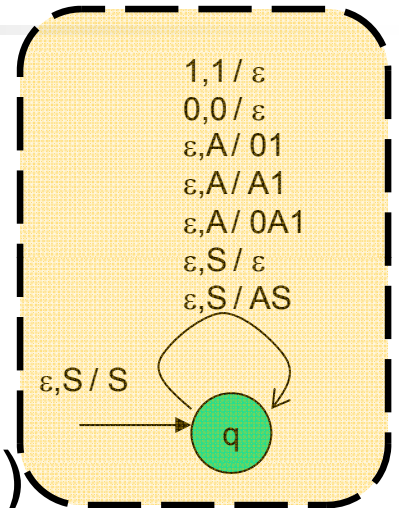
- $\delta(q, a, a) = \{ (q, \varepsilon) \}$

After:



Example: CFG to PDA

- $G = (\{S,A\}, \{0,1\}, P, S)$
- P :
 - $S \Rightarrow AS \mid \varepsilon$
 - $A \Rightarrow 0A1 \mid A1 \mid 01$
- $PDA = (\{q\}, \{0,1\}, \{0,1,A,S\}, \delta, q, S)$
- δ :
 - $\delta(q, \varepsilon, S) = \{ (q, AS), (q, \varepsilon) \}$
 - $\delta(q, \varepsilon, A) = \{ (q, 0A1), (q, A1), (q, 01) \}$
 - $\delta(q, 0, 0) = \{ (q, \varepsilon) \}$
 - $\delta(q, 1, 1) = \{ (q, \varepsilon) \}$



How will this new PDA work?

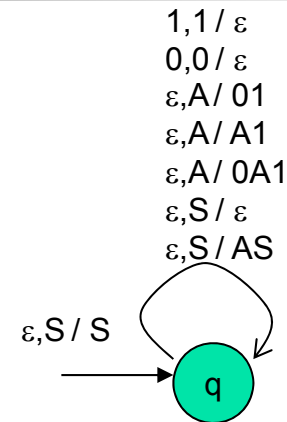
Lets simulate string 0011

Simulating string 0011 on the new PDA ...

PDA (δ):

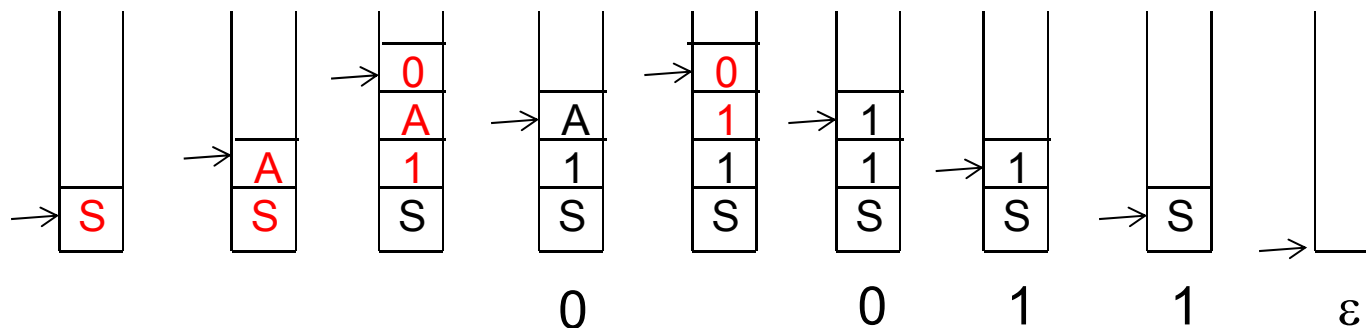
$\delta(q, \varepsilon, S) = \{ (q, AS), (q, \varepsilon) \}$
 $\delta(q, \varepsilon, A) = \{ (q, 0A1), (q, A1), (q, 01) \}$
 $\delta(q, 0, 0) = \{ (q, \varepsilon) \}$
 $\delta(q, 1, 1) = \{ (q, \varepsilon) \}$

Stack moves (shows only the successful path):



Leftmost deriv.:

$$\begin{aligned}
 S &\Rightarrow AS \\
 &\Rightarrow 0A1S \\
 &\Rightarrow 0011S \\
 &\Rightarrow 0011
 \end{aligned}$$



Accept by empty stack

$S \Rightarrow AS \Rightarrow 0A1S \Rightarrow 0011S \Rightarrow 0011$



Proof of correctness for $\text{CFG} \Rightarrow \text{PDA}$ construction

- Claim: A string is accepted by G iff it is accepted (by empty stack) by the PDA
- Proof:
 - *(only-if part)*
 - Prove by induction on the number of derivation steps
 - *(if part)*
 - If $(q, wx, S) \vdash^* (q, x, B)$ then $S \Rightarrow_{\text{Im}}^* wB$



Converting a PDA into a CFG

- Main idea: Reverse engineer the productions from transitions

If $\delta(q, a, Z) \Rightarrow (p, Y_1 Y_2 Y_3 \dots Y_k)$:

1. State is changed from q to p ;
2. Terminal a is consumed;
3. Stack top symbol Z is popped and replaced with a sequence of k variables.

- Action: Create a grammar variable called “[qZp]” which includes the following production:

- $[qZp] \Rightarrow a[pY_1q_1] [q_1Y_2q_2] [q_2Y_3q_3] \dots [q_{k-1}Y_kq_k]$

- Proof discussion (in the book)

Example: Bracket matching

- To avoid confusion, we will use $b = "("$ and $e = ")"$

$P_N: (\{q_0\}, \{b, e\}, \{Z_0, Z_1\}, \delta, q_0, Z_0)$

1. $\delta(q_0, b, Z_0) = \{(q_0, Z_1, Z_0)\}$
2. $\delta(q_0, b, Z_1) = \{(q_0, Z_1, Z_1)\}$
3. $\delta(q_0, e, Z_1) = \{(q_0, \epsilon)\}$
4. $\delta(q_0, \epsilon, Z_0) = \{(q_0, \epsilon)\}$

0. $S \Rightarrow [q_0 Z_0 q_0]$
1. $[q_0 Z_0 q_0] \Rightarrow b [q_0 Z_1 q_0] [q_0 Z_0 q_0]$
2. $[q_0 Z_1 q_0] \Rightarrow b [q_0 Z_1 q_0] [q_0 Z_1 q_0]$
3. $[q_0 Z_1 q_0] \Rightarrow e$
4. $[q_0 Z_0 q_0] \Rightarrow \epsilon$

Let $A = [q_0 Z_0 q_0]$
Let $B = [q_0 Z_1 q_0]$

0. $S \Rightarrow A$
1. $A \Rightarrow b B A$
2. $B \Rightarrow b B B$
3. $B \Rightarrow e$
4. $A \Rightarrow \epsilon$

Simplifying,

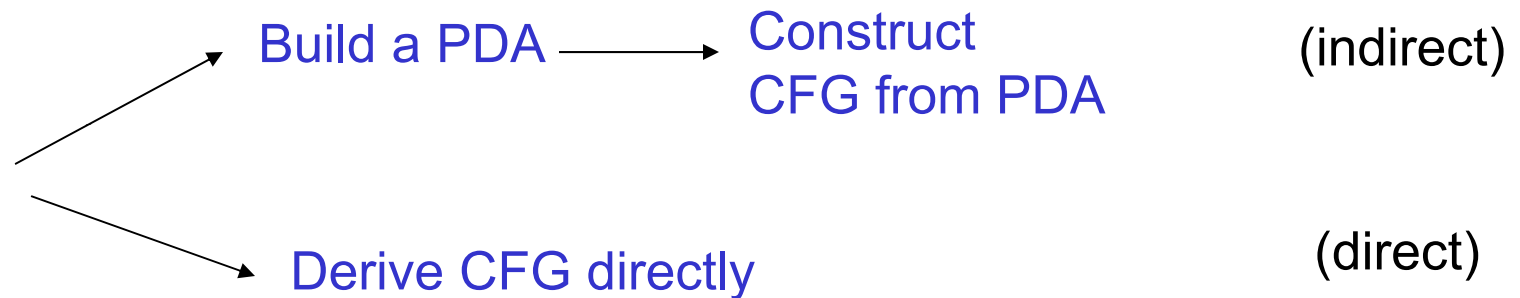
0. $S \Rightarrow b B S \mid \epsilon$
1. $B \Rightarrow b B B \mid e$

If you were to directly write a CFG:

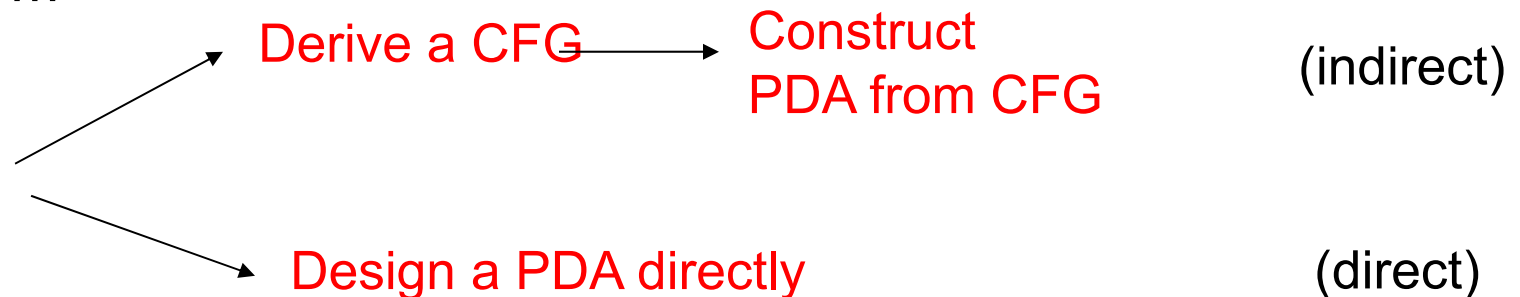
$S \Rightarrow b S e S \mid \epsilon$



Two ways to build a CFG



Similarly... Two ways to build a PDA





Deterministic PDAs

This PDA for L_{wwr} is non-deterministic

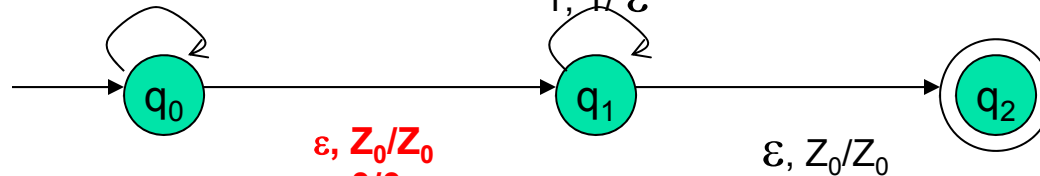
Grow stack

$0, Z_0/0Z_0$
 $1, Z_0/1Z_0$
 $0, 0/00$
 $0, 1/01$
 $1, 0/10$
 $1, 1/11$

Pop stack for
matching symbols

$0, 0/\epsilon$
 $1, 1/\epsilon$

Why does it have
to be non-
deterministic?



Switch to
popping mode

Accepts by final state

To remove
guessing,
impose the user
to insert c in the
middle

Example shows that: Nondeterministic PDAs \neq D-PDAs

D-PDA for $L_{wcw^R} = \{wcw^R \mid c \text{ is some special symbol not in } w\}$

Note:

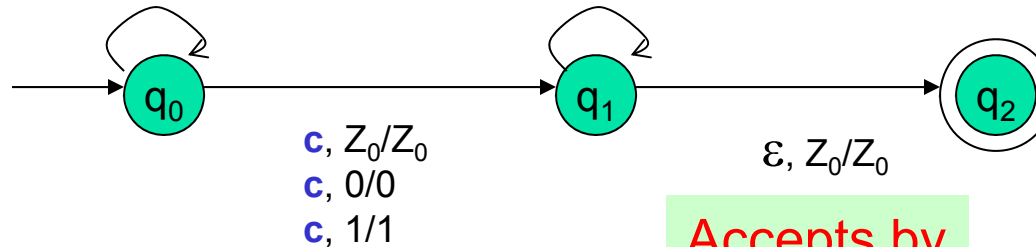
- all transitions have become deterministic

Grow stack

0, Z_0 /0 Z_0
1, Z_0 /1 Z_0
0, 0/00
0, 1/01
1, 0/10
1, 1/11

Pop stack for
matching symbols

0, 0/ ϵ
1, 1/ ϵ



Switch to
popping mode

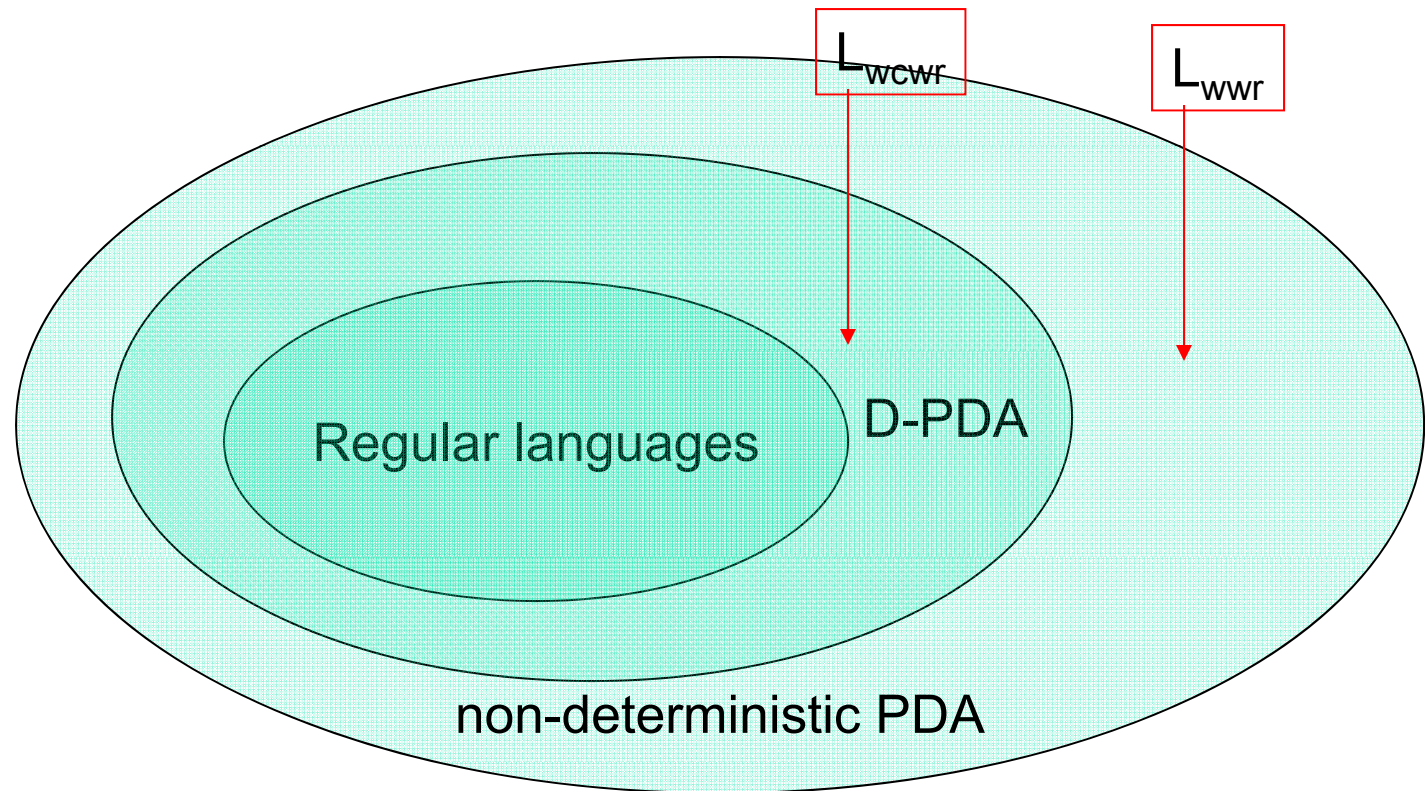
Accepts by
final state



Deterministic PDA: Definition

- A PDA is *deterministic* if and only if:
 1. $\delta(q, a, X)$ has *at most one* member for any $a \in \Sigma \cup \{\varepsilon\}$
- ➔ If $\delta(q, a, X)$ is non-empty for some $a \in \Sigma$, then $\delta(q, \varepsilon, X)$ must be empty.

PDA vs DPDA vs Regular languages





Summary

- PDAs for CFLs and CFGs
 - Non-deterministic
 - Deterministic
- PDA acceptance types
 1. By final state
 2. By empty stack
- PDA
 - IDs, Transition diagram
- Equivalence of CFG and PDA
 - $\text{CFG} \Rightarrow \text{PDA}$ construction
 - $\text{PDA} \Rightarrow \text{CFG}$ construction