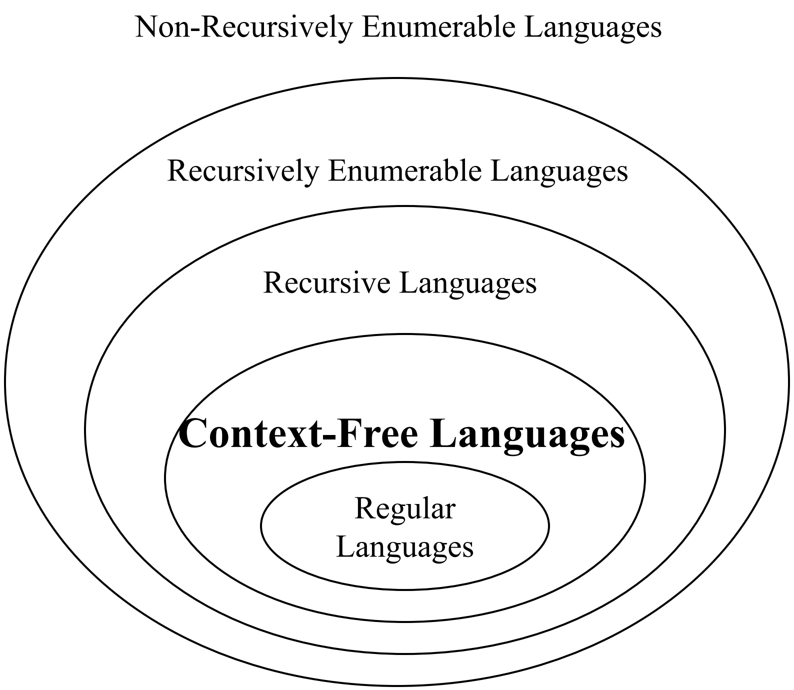
Hierarchy of languages

Regular Languages Finite State Machines, Regular Expression

Context Free Languages Context Free Grammar, **Push-down Automata**



Pushdown Automata (PDA)

- **Informally:**
  - A PDA is an NFA-ε with a stack.
  - Transitions are modified to accommodate stack operations.

- **Questions:**
  - What is a stack?
  - How does a stack help?

- A DFA can "remember" only a finite amount of information, whereas a PDA can "remember" an infinite amount of (certain types of) information, in one memory-stack

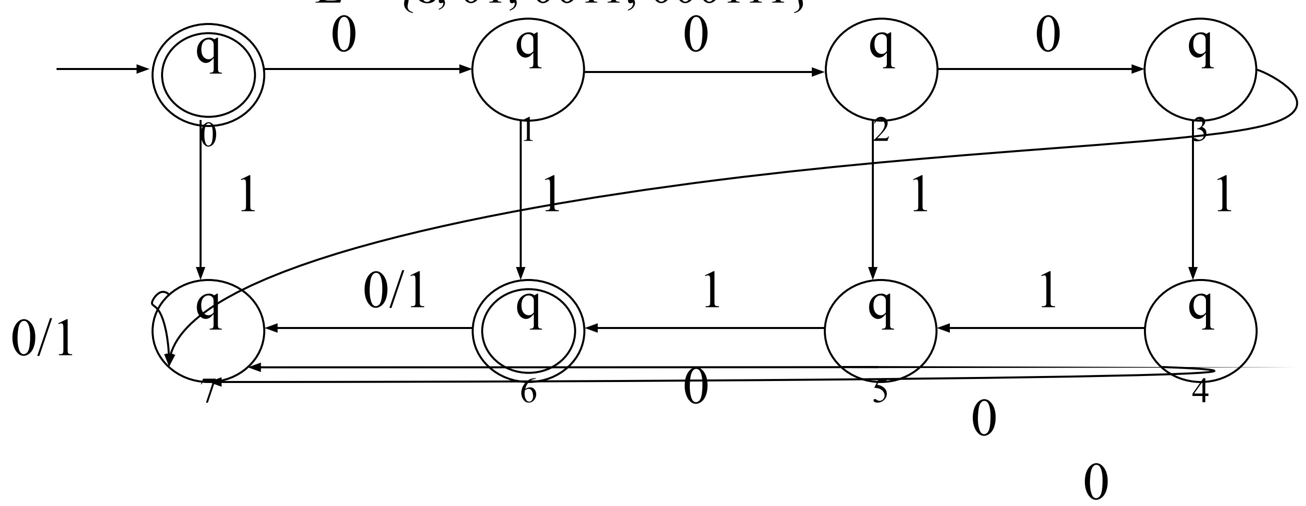
- **Example:**

  $\{0^n1^n \mid 0 =< n\}$ is *not* regular, but

  $\{0^n1^n \mid 0 \le n \le k$, for some fixed $k\}$ is regular, for any fixed k.

- **For k=3:**

  L = {ε, 01, 0011, 000111}



- In a DFA, each state remembers a finite amount of information.

- To get $\{0^n 1^n \mid 0 \leq n\}$ with a DFA would require an infinite number of states using the preceding technique.

- An infinite stack solves the problem for $\{0^n 1^n \mid 0 \leq n\}$ as follows:
  - Read all 0's and place them on a stack
  - Read all 1's and match with the corresponding 0's on the stack

- Only need two states to do this in a PDA

- Similarly for $\{0^n 1^m 0^{n+m} \mid n,m \geq 0\}$

Formal Definition of a PDA

- A <u>pushdown automaton (PDA)</u> is a seven-tuple:

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

Q	A <u>finite</u> set of states
Σ	A <u>finite</u> input alphabet
Γ	A <u>finite</u> stack alphabet
q_0	The initial/starting state, q_0 is in Q
z_0	A starting stack symbol, is in Γ // need not always remain at the bottom of stack
F	A set of final/accepting states, which is a subset of Q
δ	A transition function, where

δ: Q x (Σ U {ε}) x Γ –> finite subsets of Q x Γ^*

- Consider the various parts of δ:

  $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to$ finite subsets of $Q \times \Gamma^*$

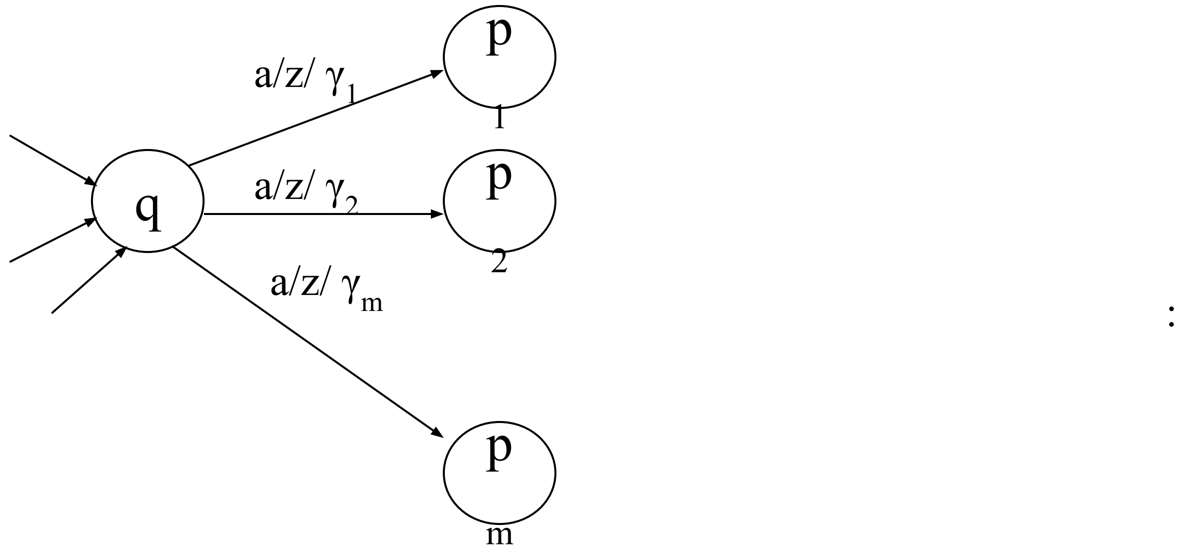
  - Q on the LHS means that at each step in a computation, a PDA must consider its' current state.
  - Γ on the LHS means that at each step in a computation, a PDA must consider the symbol on top of its' stack.
  - $\Sigma \cup \{\varepsilon\}$ on the LHS means that at each step in a computation, a PDA may or may not consider the current input symbol, i.e., it may have epsilon transitions.

  - "Finite subsets" on the RHS means that at each step in a computation, a PDA may have several options.
  - Q on the RHS means that each option specifies a new state.
  - Γ^* on the RHS means that each option specifies zero or more stack symbols that will replace the top stack symbol, but *in a specific sequence*.

- **Two types of PDA transitions:**

$$\delta(q, a, z) = \{(p_1,\gamma_1), (p_2,\gamma_2),\ldots, (p_m,\gamma_m)\}$$

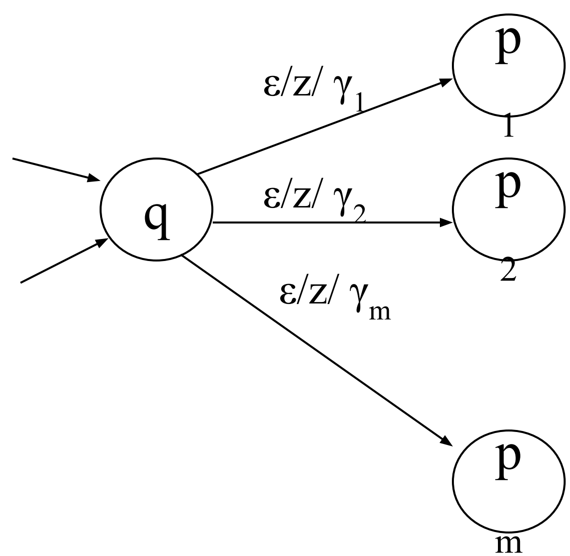
- Current state is q
- Current input symbol is a
- Symbol currently on top of the stack z
- Move to state p_i from q
- Replace z with γ_i on the stack (leftmost symbol on top)
- Move the input head to the next input symbol



- **Two types of PDA transitions:**

$$\delta(q, \varepsilon, z) = \{(p_1,\gamma_1), (p_2,\gamma_2),\ldots, (p_m,\gamma_m)\}$$

- Current state is q
- Current input symbol is not considered
- Symbol currently on top of the stack z
- Move to state p_i from q
- Replace z with γ_i on the stack (leftmost symbol on top)
- **No input symbol is read**



:

- **Example:** 0^n1^n, n>=0

  $M = (\{q_1, q_2\}, \{0, 1\}, \{L, \#\}, \delta, q_1, \#, \varnothing)$

  δ:

  (1) $\delta(q_1, 0, \#) = \{(q_1, L\#)\}$ // stack order: L on top, then # below

  (2) $\delta(q_1, 1, \#) = \varnothing$ // illegal, string rejected, *When will it happen?*

  (3) $\delta(q_1, 0, L) = \{(q_1, LL)\}$

  (4) $\delta(q_1, 1, L) = \{(q_2, \varepsilon)\}$

  (5) $\delta(q_2, 1, L) = \{(q_2, \varepsilon)\}$

  (6) $\delta(q_2, \varepsilon, \#) = \{(q_2, \varepsilon)\}$ //if ε read & stack hits bottom, accept

  (7) $\delta(q_2, \varepsilon, L) = \varnothing$ // illegal, string rejected

  (8) $\delta(q_1, \varepsilon, \#) = \{(q_2, \varepsilon)\}$ // n=0, accept

- **Goal:** (acceptance)
  - Read the entire input string
  - Terminate with an empty stack

- Informally, a string is accepted if there exists a computation that uses up all the input and leaves the stack empty.
- *How many rules should be there in delta?*

- **Language: 0^n1^n, n>=0**

δ:

(1) $\delta(q_1, 0, \#) = \{(q_1, L\#)\}$ **// stack order: L on top, then # below**

(2) $\delta(q_1, 1, \#) = \emptyset$ **// illegal, string rejected, *When will it happen?***

(3) $\delta(q_1, 0, L) = \{(q_1, LL)\}$

(4) $\delta(q_1, 1, L) = \{(q_2, \varepsilon)\}$

(5) $\delta(q_2, 1, L) = \{(q_2, \varepsilon)\}$

(6) $\delta(q_2, \varepsilon, \#) = \{(q_2, \varepsilon)\}$ **//if ε read & stack hits bottom, accept**

(7) $\delta(q_2, \varepsilon, L) = \emptyset$ **// illegal, string rejected**

(8) $\delta(q_1, \varepsilon, \#) = \{(q_2, \varepsilon)\}$ **// n=0, accept**

- **0011**
- ***(q1, 0 011, #) |-***
  (q1, 0 11, L#) |-
  (q1, 1 1, LL#) |-
  (q2, 1, L#) |-
  (q2, e, #) |-
  (q2, e, e): accept
- ***011***
- ***(q1, 0 11, #) |-***
  (q1, 1 1, L#) |-
  (q2, 1, #) |-
  Ø : reject
- ***Try 001***

- **Example:** balanced parentheses,
- e.g. in-language: ((())()), or (())(), but not-in-language: ((())

$M = (\{q_1\}, \{``(", ``)"\}, \{L, \#\}, \delta, q_1, \#, \varnothing)$

δ:

(1) $\delta(q_1, (, \#) = \{(q_1, L\#)\}$ // stack order: L-on top-then- # lower

(2) $\delta(q_1,), \#) = \varnothing$ // illegal, string rejected

(3) $\delta(q_1, (, L) = \{(q_1, LL)\}$

(4) $\delta(q_1,), L) = \{(q_1, \varepsilon)\}$

(5) $\delta(q_1, \varepsilon, \#) = \{(q_1, \varepsilon)\}$ //if ε read & stack hits bottom, accept

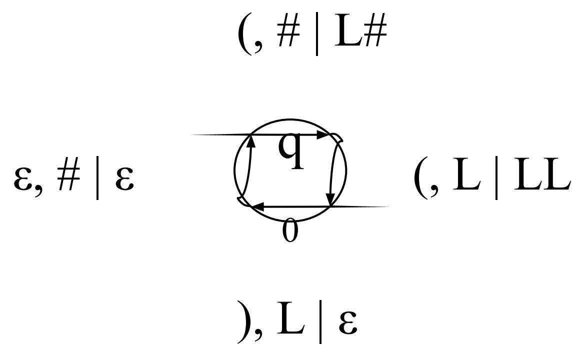
(6) $\delta(q_1, \varepsilon, L) = \varnothing$ // illegal, string rejected

// *What does it mean? When will it happen?*

- **Goal:** (acceptance)
  - Read the entire input string
  - Terminate with an empty stack

- Informally, a string is accepted if there exists a computation that uses up all the input and leaves the stack empty.
- *How many rules should be in delta?*

- Transition Diagram:



- Example Computation:

Current Input	Stack	Transition
(())	#	-- initial status
())	L#	(1)
- Could have applied rule (5), but		
))	LL#	(3)
it would have done no good		
)	L#	(4)
ε	#	(4)
ε	-	(5)

- **Example PDA #1:** For the language $\{x \mid x = wcw^r$ and w in $\{0,1\}^*$, but sigma=$\{0,1,c\}\}$
- *Is this a regular language?*
- *Note: length |x| is odd*

$M = (\{q_1, q_2\}, \{0, 1, c\}, \{\#, B, G\}, \delta, q_1, \#, \emptyset)$

δ:

(1) $\delta(q_1, 0, \#) = \{(q_1, B\#)\}$ (9) $\delta(q_1, 1, \#) = \{(q_1,$
$G\#)\}$

(2) $\delta(q_1, 0, B) = \{(q_1, BB)\}$ (10) $\delta(q_1, 1, B) = \{(q_1, GB)\}$

(3) $\delta(q_1, 0, G) = \{(q_1, BG)\}$ (11) $\delta(q_1, 1, G) = \{(q_1, GG)\}$

(4) $\delta(q_1, c, \#) = \{(q_2, \#)\}$

(5) $\delta(q_1, c, B) = \{(q_2, B)\}$

(6) $\delta(q_1, c, G) = \{(q_2, G)\}$

(7) $\delta(q_2, 0, B) = \{(q_2, \varepsilon)\}$ (12) $\delta(q_2, 1, G) = \{(q_2, \varepsilon)\}$

(8) $\delta(q_2, \varepsilon, \#) = \{(q_2, \varepsilon)\}$

- **Notes:**
  - Stack grows leftwards
  - Only rule #8 is non-deterministic.
  - Rule #8 is used to pop the final stack symbol off at the end of a computation.

- **Example Computation:**

(1)	$\delta(q_1, 0, \#) = \{(q_1, B\#)\}$	(9)	$\delta(q_1, 1, \#) = \{(q_1, G\#)\}$
(2)	$\delta(q_1, 0, B) = \{(q_1, BB)\}$	(10)	$\delta(q_1, 1, B) = \{(q_1, GB)\}$
(3)	$\delta(q_1, 0, G) = \{(q_1, BG)\}$	(11)	$\delta(q_1, 1, G) = \{(q_1, GG)\}$
(4)	$\delta(q_1, c, \#) = \{(q_2, \#)\}$		
(5)	$\delta(q_1, c, B) = \{(q_2, B)\}$		
(6)	$\delta(q_1, c, G) = \{(q_2, G)\}$		
(7)	$\delta(q_2, 0, B) = \{(q_2, \varepsilon)\}$	(12)	$\delta(q_2, 1, G) = \{(q_2, \varepsilon)\}$
(8)	$\delta(q_2, \varepsilon, \#) = \{(q_2, \varepsilon)\}$		

State	Input	Stack	Rule Applied	Rules Applicable
q_1	**0**1c10	#		(1)
q_1	**1**c10	B#	(1)	(10)
q_1	**c**10	GB#	(10)	(6)
q_2	**1**0	GB#	(6)	(12)

- **Example Computation:**

(1) $\delta(q_1, 0, \#) = \{(q_1, B\#)\}$

(2) $\delta(q_1, 0, B) = \{(q_1, BB)\}$

(3) $\delta(q_1, 0, G) = \{(q_1, BG)\}$

(4) $\delta(q_1, c, \#) = \{(q_2, \#)\}$

(5) $\delta(q_1, c, B) = \{(q_2, B)\}$

(6) $\delta(q_1, c, G) = \{(q_2, G)\}$

(7) $\delta(q_2, 0, B) = \{(q_2, \varepsilon)\}$

(8) $\delta(q_2, \varepsilon, \#) = \{(q_2, \varepsilon)\}$

(9) $\delta(q_1, 1, \#) = \{(q_1, G\#)\}$

(10) $\delta(q_1, 1, B) = \{(q_1, GB)\}$

(11) $\delta(q_1, 1, G) = \{(q_1, GG)\}$

(12) $\delta(q_2, 1, G) = \{(q_2, \varepsilon)\}$

State	Input	Stack	Rule Applied
q_1	**1**c1	#	
q_1	**c**1	G#	(9)
q_2	**1**	G#	(6)
q_2	**ε**	#	(12)
q_2	ε	ε	(8)

- **Questions:**
  - Why isn't $\delta(q_2, 0, G)$ defined?
  - Why isn't $\delta(q_2, 1, B)$ defined?
- TRY: 11c1

- **Example PDA #2:** For the language $\{x \mid x = ww^r \text{ and } w \text{ in } \{0,1\}^*\}$
- *Note: length |x| is even*

$M = (\{q_1, q_2\}, \{0, 1\}, \{\#, B, G\}, \delta, q_1, \#, \emptyset)$

δ:

(1) $\delta(q_1, 0, \#) = \{(q_1, B\#)\}$

(2) $\delta(q_1, 1, \#) = \{(q_1, G\#)\}$

(3) $\delta(q_1, 0, B) = \{(q_1, BB), (q_2, \varepsilon)\}$

(4) $\delta(q_1, 0, G) = \{(q_1, BG)\}$

(5) $\delta(q_1, 1, B) = \{(q_1, GB)\}$

(6) $\delta(q_1, 1, G) = \{(q_1, GG), (q_2, \varepsilon)\}$

(7) $\delta(q_2, 0, B) = \{(q_2, \varepsilon)\}$

(8) $\delta(q_2, 1, G) = \{(q_2, \varepsilon)\}$

(9) $\delta(q_1, \varepsilon, \#) = \{(q_2, \#)\}$

(10) $\delta(q_2, \varepsilon, \#) = \{(q_2, \varepsilon)\}$

- **Notes:**
  - Rules #3 and #6 are non-deterministic: two options each
  - Rules #9 and #10 are used to pop the final stack symbol off at the end of a computation.

- **Example Computation:**

(1) $\delta(q_1, 0, \#) = \{(q_1, B\#)\}$

(2) $\delta(q_1, 1, \#) = \{(q_1, G\#)\}$

(3) $\delta(q_1, 0, B) = \{(q_1, BB), (q_2, \varepsilon)\}$

(4) $\delta(q_1, 0, G) = \{(q_1, BG)\}$

(5) $\delta(q_1, 1, B) = \{(q_1, GB)\}$

(6) $\delta(q_1, 1, G) = \{(q_1, GG), (q_2, \varepsilon)\}$

(7) $\delta(q_2, 0, B) = \{(q_2, \varepsilon)\}$

(8) $\delta(q_2, 1, G) = \{(q_2, \varepsilon)\}$

(9) $\delta(q_1, \varepsilon, \#) = \{(q_2, \varepsilon)\}$

(10) $\delta(q_2, \varepsilon, \#) = \{(q_2, \varepsilon)\}$

State	Input	Stack	Rule Applied	Rules Applicable
q_1	000000	#		(1), (9)
q_1	00000	B#	(1)	(3), both options
q_1	0000	BB#	(3) option #1	(3), both options
q_1	000	BBB#	(3) option #1	(3), both options
q_2	00	BB#	(3) option #2	(7)
q_2	0	B#	(7)	(7)
q_2	ε	#	(7)	(10)
q_2	ε	ε	(10)	

- **Negative Example Computation:**

(1) $\delta(q_1, 0, \#) = \{(q_1, B\#)\}$
(2) $\delta(q_1, 1, \#) = \{(q_1, G\#)\}$
(3) $\delta(q_1, 0, B) = \{(q_1, BB), (q_2, \varepsilon)\}$
(4) $\delta(q_1, 0, G) = \{(q_1, BG)\}$
(5) $\delta(q_1, 1, B) = \{(q_1, GB)\}$
(6) $\delta(q_1, 1, G) = \{(q_1, GG), (q_2, \varepsilon)\}$
(7) $\delta(q_2, 0, B) = \{(q_2, \varepsilon)\}$
(8) $\delta(q_2, 1, G) = \{(q_2, \varepsilon)\}$
(9) $\delta(q_1, \varepsilon, \#) = \{(q_2, \varepsilon)\}$
(10) $\delta(q_2, \varepsilon, \#) = \{(q_2, \varepsilon)\}$

State	Input	Stack	Rule Applied
q_1	000	#	
q_1	00	B#	(1)
q_1	0	BB#	(3) option #1

(q2, 0, #) by option 2

q_1	ε	BBB#	(3) option #1

-crashes, no-rule to apply-

(q2, ε, B#) by option 2

-rejects: end of string but not empty stack-

- **Example Computation:**

(1)	$\delta(q_1, 0, \#) = \{(q_1, B\#)\}$	(6)	$\delta(q_1, 1, G) = \{(q_1, GG), (q_2, \varepsilon)\}$
(2)	$\delta(q_1, 1, \#) = \{(q_1, G\#)\}$	(7)	$\delta(q_2, 0, B) = \{(q_2, \varepsilon)\}$
(3)	$\delta(q_1, 0, B) = \{(q_1, BB), (q_2, \varepsilon)\}$	(8)	$\delta(q_2, 1, G) = \{(q_2, \varepsilon)\}$
(4)	$\delta(q_1, 0, G) = \{(q_1, BG)\}$	(9)	$\delta(q_1, \varepsilon, \#) = \{(q_2, \varepsilon)\}$
(5)	$\delta(q_1, 1, B) = \{(q_1, GB)\}$	(10)	$\delta(q_2, \varepsilon, \#) = \{(q_2, \varepsilon)\}$

State	Input	Stack	Rule Applied
q_1	010010	#	
q_1	10010	B#	(1) From (1) and (9)
q_1	0010	GB#	(5)
q_1	010	BGB#	(4)
q_2	10	GB#	(3) option #2
q_2	0	B#	(8)
q_2	ε	#	(7)
q_2	ε	ε	(10)

- **Exercises:**
  - 0011001100 // *how many total options the machine (or you!) may need to try before rejection?*
  - 011110
  - 0111

Formal Definitions for PDAs

- Let $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA.

- **Definition:** An *instantaneous description* (ID) is a triple (q, w, γ), where q is in Q, w is in Σ^* and γ is in Γ^*.
  - q is the current state
  - w is the unused input
  - γ is the current stack contents

- **Example:** (for PDA #2)

$(q_1, 111, GBR)$	$(q_1, 11, GGBR)$
$(q_1, 111, GBR)$	$(q_2, 11, BR)$
$(q_1, 000, GR)$	$(q_2, 00, R)$

- Let $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA.

- **Definition:** Let a be in $\Sigma \cup \{\varepsilon\}$, w be in Σ^*, z be in Γ, and α and β both be in Γ^*. Then:

$$(q, aw, z\alpha) \vdash_M (p, w, \beta\alpha)$$

  if $\delta(q, a, z)$ contains (p, β).

- Intuitively, if I and J are instantaneous descriptions, then $I \vdash J$ means that J follows from I by one transition.

- **Examples:** (PDA #2)

$(q_1, 111, GBR) \vdash (q_1, 11, GGBR)$ (6) option #1, with a=1, z=G, β=GG, w=11, and α= BR

$(q_1, 111, GBR) \vdash (q_2, 11, BR)$ (6) option #2, with a=1, z=G, β= ε, w=11, and α= BR

$(q_1, 000, GR) \vdash (q_2, 00, R)$ Is *not* true, For any a, z, β, w and α

- **Examples:** (PDA #1)

$(q_1, (())), L\#) \vdash (q_1, ())),LL\#)$ (3)

- **Definition:** |—* is the reflexive and transitive closure of |—.
  - I |—* I for each instantaneous description I
  - If I |— J and J |—* K then I |—* K

- Intuitively, if I and J are instantaneous descriptions, then I |—* J means that J follows from I by zero or more transitions.

- **Definition:** Let $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA. The *language accepted by empty stack*, denoted $L_E(M)$, is the set

$$\{w \mid (q_0, w, z_0) \vdash^* (p, \varepsilon, \varepsilon) \text{ for some } p \text{ in } Q\}$$

- **Definition:** Let $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA. The *language accepted by final state*, denoted $L_F(M)$, is the set

$$\{w \mid (q_0, w, z_0) \vdash^* (p, \varepsilon, \gamma) \text{ for some } p \text{ in } F \text{ and } \gamma \text{ in } \Gamma^*\}$$

- **Definition:** Let $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA. The *language accepted by empty stack and final state*, denoted $L(M)$, is the set

$$\{w \mid (q_0, w, z_0) \vdash^* (p, \varepsilon, \varepsilon) \text{ for some } p \text{ in } F\}$$

- **Lemma 1:** Let $L = L_E(M_1)$ for some PDA M_1. Then there exits a PDA M_2 such that $L = L_F(M_2)$.

- **Lemma 2:** Let $L = L_F(M_1)$ for some PDA M_1. Then there exits a PDA M_2 such that $L = L_E(M_2)$.

- **Theorem:** Let L be a language. Then there exits a PDA M_1 such that $L = L_F(M_1)$ if and only if there exists a PDA M_2 such that $L = L_E(M_2)$.

- **Corollary:** The PDAs that accept by empty stack and the PDAs that accept by final state define the same class of languages.

- **Note:** Similar lemmas and theorems could be stated for PDAs that accept by both final state and empty stack.

Back to CFG again:
PDA equivalent to CFG

- **Definition:** Let G = (V, T, P, S) be a CFL. If every production in P is of the form

$$A \rightarrow a\alpha$$

Where A is in V, *a* is in T, and α is in V*, then G is said to be in <u>Greibach Normal</u> <u>Form</u> (GNF).

Only one non-terminal in front.

- **Example:**

$S \rightarrow aAB \mid bB$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid c$ *Language:* $(aa^{+}+b)b^{+}c$

- **Theorem:** Let L be a CFL. Then L – {ε} is a CFL.

- **Theorem:** Let L be a CFL not containing {ε}. Then there exists a GNF grammar G such that L = L(G).

- **Lemma 1:** Let *L* be a CFL. Then there exists a PDA *M* such that $L = L_E(M)$.

- **Proof:** Assume without loss of generality that ε is not in *L*. The construction can be modified to include ε later.

  Let G = (V, T, P, S) be a CFG, and assume without loss of generality that *G* is in GNF. Construct M = (Q, Σ, Γ, δ, q, z, Ø) where:

  Q = {q}

  Σ = T

  Γ = V

  z = S

  δ: for all *a* in Σ and *A* in Γ, δ(q, a, A) contains (q, γ)

  if $A \rightarrow a\gamma$ is in *P* or rather:

  $\delta(q, a, A) = \{(q, \gamma) \mid A \rightarrow a\gamma$ is in *P* and γ is in Γ*},

  for all *a* in Σ and *A* in Γ

- For a given string *x* in Σ* , *M* will attempt to simulate a leftmost derivation of *x* with *G*.

- **Example #1:** Consider the following CFG in GNF.

$S \rightarrow aS$ G is in GNF

$S \rightarrow a$ $L(G) = a^+$

Construct M as:

$Q = \{q\}$

$\Sigma = T = \{a\}$

$\Gamma = V = \{S\}$

$z = S$

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

$\delta(q, \varepsilon, S) = \emptyset$

- *Is δ complete?*

- **Example #2:** Consider the following CFG in GNF.

(1) S –> aA
(2) S –> aB
(3) A –> aA G is in GNF
(4) A –> aB $L(G) = a^+ b^+$ // This looks ok to me, one, two or more *a*'s in the start
(5) B –> bB
(6) B –> b *[Can you write a simpler equivalent CFG? Will it be GNF?]*

Construct M as:
Q = {q}
Σ = T = {a, b}
Γ = V = {S, A, B}
z = S

(1) δ(q, a, S) = {(q, A), (q, B)} From productions #1 and 2, S->aA, S->aB
(2) δ(q, a, A) = {(q, A), (q, B)} From productions #3 and 4, A->aA, A->aB
(3) δ(q, a, B) = Ø
(4) δ(q, b, S) = Ø
(5) δ(q, b, A) = Ø
(6) δ(q, b, B) = {(q, B), (q, ε)} From productions #5 and 6, B->bB, B->b
(7) δ(q, ε, S) = Ø
(8) δ(q, ε, A) = Ø
(9) δ(q, ε, B) = Ø *Is δ complete?*

- For a string w in L(G) the PDA M will simulate a leftmost derivation of w.
  - If w is in L(G) then $(q, w, z_0) \vdash^* (q, \varepsilon, \varepsilon)$
  - If $(q, w, z_0) \vdash^* (q, \varepsilon, \varepsilon)$ then w is in L(G)

- Consider generating a string using G. Since G is in GNF, each sentential form in a *leftmost* derivation has form:

$$\Rightarrow t_1 t_2 \ldots t_i A_1 A_2 \ldots A_m$$

terminals non-terminals

- And each step in the derivation (i.e., each application of a production) adds a terminal and some non-terminals.

$$A_1 \rightarrow t_{i+1} \alpha$$

$$\Rightarrow t_1 t_2 \ldots t_i t_{i+1} \alpha A_1 A_2 \ldots A_m$$

- Each transition of the PDA simulates one derivation step. Thus, the i^{th} step of the PDAs' computation corresponds to the i^{th} step in a corresponding leftmost derivation with the grammar.

- After the i^{th} step of the computation of the PDA, $t_1 t_2 \ldots t_{i+1}$ are the symbols that have already been read by the PDA and $\alpha A_1 A_2 \ldots A_m$ are the stack contents.

- For each leftmost derivation of a string generated by the grammar, there is an equivalent accepting computation of that string by the PDA.
- Each sentential form in the leftmost derivation corresponds to an instantaneous description in the PDA's corresponding computation.
- For example, the PDA instantaneous description corresponding to the sentential form:

$$\Rightarrow t_1 t_2 \ldots t_i A_1 A_2 \ldots A_m$$

would be:

$$(q,\ t_{i+1} t_{i+2} \ldots t_n,\ A_1 A_2 \ldots A_m)$$

- **Example:** Using the grammar from example #2:

S	=> aA	(1)
	=> aaA	(3)
	=> aaaA	(3)
	=> aaaaB	(4)
	=> aaaabB	(5)
	=> aaaabb	(6)

Grammar:

(1) S –> aA
(2) S –> aB
(3) A –> aA
G is in GNF
(4) A –> aB
$L(G) = a^+b^+$
(5) B –> bB
(6) B –> b

- The corresponding computation of the PDA:

(1) δ(q, a, S) = {(q, A), (q, B)}

(2) δ(q, a, A) = {(q, A), (q, B)}

(3) δ(q, a, B) = Ø

(4) δ(q, b, S) = Ø

(5) δ(q, b, A) = Ø

(6) δ(q, b, B) = {(q, B), (q, ε)}

(7) δ(q, ε, S) = Ø

(8) δ(q, ε, A) = Ø

(9) δ(q, ε, B) = Ø

(rule#)/right-side#

- (q, aaaabb, S) |— (q, aaabb, A) (1)/1
  |— (q, aabb, A) (2)/1
  |— (q, abb, A) (2)/1
  |— (q, bb, B) (2)/2
  |— (q, b, B) (6)/1
  |— (q, ε, ε) (6)/2

  – String is read
  – Stack is emptied
  – Therefore the string is accepted by the PDA

- **Another Example:** Using the PDA from example #2:

(q, aabb, S)	$\vdash$ (q, abb, A)	(1)/1
	$\vdash$ (q, bb, B)	(2)/2
	$\vdash$ (q, b, B)	(6)/1
	$\vdash$ (q, ε, ε)	(6)/2

- The corresponding derivation using the grammar:

S	=> aA	(1)
	=> aaB	(4)
	=> aabB	(5)
	=> aabb	(6)

- **Example #3:** Consider the following CFG in GNF.

(1) S –> aABC
(2) A –> a
(3) B –> b
(4) C –> cAB
(5) C –> cC

G is in GNF

Language?

$aab\ cc^{*}\ ab$

Construct M as:

Q = {q}
Σ = T = {a, b, c}
Γ = V = {S, A, B, C}
z = S

(1) δ(q, a, S) = {(q, ABC)} S->aABC
(2) δ(q, a, A) = {(q, ε)} A->a
(3) δ(q, a, B) = Ø
(4) δ(q, a, C) = Ø
(5) δ(q, b, S) = Ø
(6) δ(q, b, A) = Ø
(7) δ(q, b, B) = {(q, ε)} B->b

(9) δ(q, c, S) = Ø
(10) δ(q, c, A) = Ø
(11) δ(q, c, B) = Ø
(12) δ(q, c, C) = {(q, AB), (q, C))} // C->cAB|cC
(13) δ(q, ε, S) = Ø
(14) δ(q, ε, A) = Ø
(15) δ(q, ε, B) = Ø

- **Notes:**
  - Recall that the grammar *G* was required to be in GNF before the construction could be applied.
  - As a result, it was assumed at the start that ε was not in the context-free language *L*.
  - What if ε is in *L*? You need to add ε back.

- **Suppose ε is in L:**

1) First, let L' = L – {ε}

Fact: If *L* is a CFL, then L' = L – {ε} is a CFL.

By an earlier theorem, there is GNF grammar *G* such that L' = L(G).

2) Construct a PDA *M* such that L' = L_E(M)

How do we modify *M* to accept ε?

Add δ(q, ε, S) = {(q, ε)}? *NO!!*

- **Counter Example:**

  Consider L = {ε, b, ab, aab, aaab, …}= ε + a*b Then L' = {b, ab, aab, aaab, …} = a*b

- **The GNF CFG for L':**

P:

(1) S –> aS

(2) S –> b

- **The PDA M Accepting L':**

  Q = {q}

  Σ = T = {a, b}

  Γ = V = {S}

  z = S

  δ(q, a, S) = {(q, S)}

  δ(q, b, S) = {(q, ε)}

  δ(q, ε, S) = Ø

How to add ε to L' now?

$\delta(q, a, S) = \{(q, S)\}$

$\delta(q, b, S) = \{(q, \varepsilon)\}$

$\delta(q, \varepsilon, S) = \emptyset$

- If $\delta(q, \varepsilon, S) = \{(q, \varepsilon)\}$ is added then:

  $L(M) = \{\varepsilon, a, aa, aaa, \ldots, b, ab, aab, aaab, \ldots\}$, wrong!

  It is like, S -> aS | b | ε

  which is wrong!

  Correct grammar should be:

  (0) $S_1 \rightarrow \varepsilon \mid S$, with new starting non-terminal S_1

  (1) S –> aS

  (2) S –> b

For PDA, add a new *Stack-bottom symbol* S_1, with new transitions:

$\delta(q, \varepsilon, S_1) = \{(q, \varepsilon), (q, S)\}$, where *S* was the previous stack-bottom of *M*

Alternatively, add a new *start* state q' with transitions:

$\delta(q', \varepsilon, S) = \{(q', \varepsilon), (q, S)\}$

- **Lemma 1:** Let *L* be a CFL. Then there exists a PDA *M* such that $L = L_E(M)$.

- **Lemma 2:** Let *M* be a PDA. Then there exists a CFG grammar *G* such that $L_E(M) = L(G)$.
  - *Can you prove it?*
  - *First step would be to transform an arbitrary PDA to a single state PDA!*

- **Theorem:** Let *L* be a language. Then there exists a CFG *G* such that L = L(G) iff there exists a PDA *M* such that $L = L_E(M)$.

- **Corollary:** The PDAs define the CFLs.

Sample CFG to GNF transformation:

- $0^n 1^n$, n>=1
- S -> 0S1 | 01

- GNF:
- S -> $0SS_1$ | $0S_1$
- S_1 -> 1

- *Note: in PDA the symbol S will float on top, rather than stay at the bottom!*
- *Acceptance of string by removing last S_1 at stack bottom*

Ignore this slide

- How about language like: ((())()))(), nested

$M = (\{q_1, q_2\}, \{\text{“(”}, \text{“)”}\}, \{L, \#\}, \delta, q_1, \#, \varnothing)$

δ:

(1) $\delta(q_1, (, \#) = \{(q_1, L\#)\}$

(2) $\delta(q_1,), \#) = \varnothing$ // illegal, string rejected

(3) $\delta(q_1, (, L) = \{(q_1, LL)\}$

(4) $\delta(q_1,), L) = \{(q_2, \varepsilon)\}$

(5) $\delta(q_2,), L) = \{(q_2, \varepsilon)\}$

(6) $\delta(q_2, (, L) = \{(q_1, LL)\}$ // not balanced yet, but start back anyway

(7) $\delta(q_2, (, \#) = \{(q_1, L\#)\}$ // start afresh again

(8) $\delta(q_2, \varepsilon, \#) = \{(q_2, \varepsilon)\}$ // end of string & stack hits bottom, accept

(9) $\delta(q_1, \varepsilon, \#) = \{(q_1, \varepsilon)\}$ // special rule for empty string

(10) $\delta(q_1, \varepsilon, L) = \varnothing$ // illegal, end of string but more L in stack

Total number of transitions? Verify all carefully.