# Pushdown Automata

Reading: Chapter 6

# Pushdown Automata (PDA)

- **Informally:**
  - A PDA is an NFA-ε with a <u>infinite</u> 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.

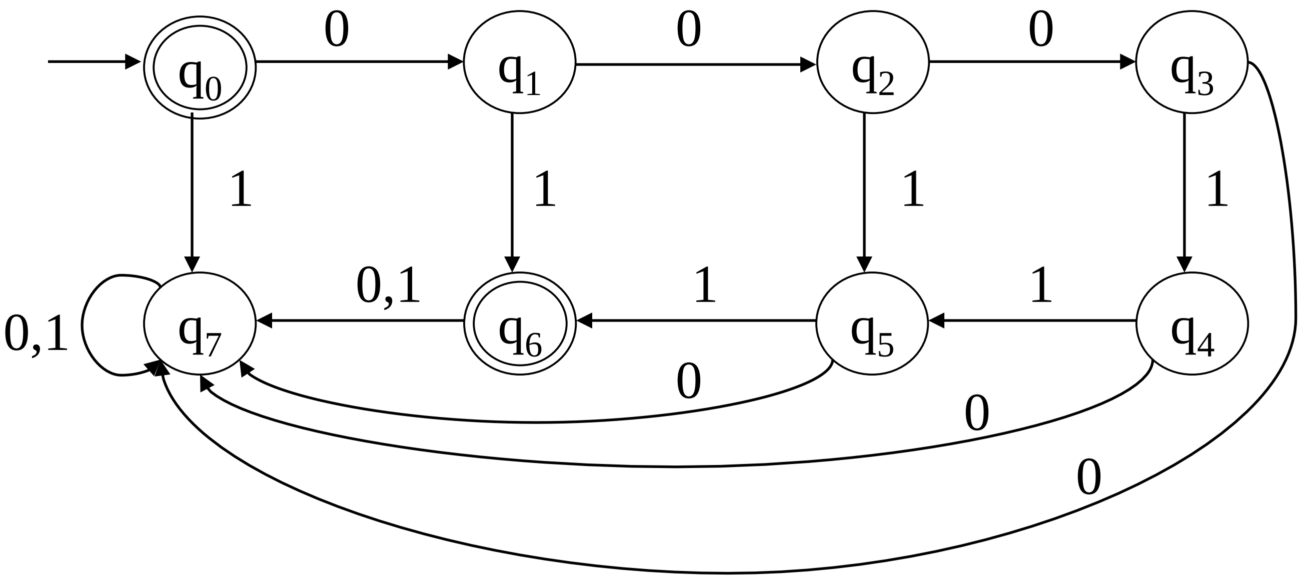
- **Example:**

  $\{0^n1^n \mid 0=<n\}$ Is *not* regular

  $\{0^n1^n \mid 0=<n<=k$, 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 $\{0^n1^n \mid n>=0\}$ with a DFA would require an infinite number of states using the preceding technique.

- An infinite stack solves the problem for $\{0^n1^n \mid 0=<n\}$ 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 $\{0^n1^m0^{n+m} \mid n,m>=0\}$

# 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 |
| $\Sigma$ | A <u>finite</u> input alphabet |
| $\Gamma$ | 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 $\Gamma$ |
| $F$ | A set of final/accepting states, which is a subset of Q |
| $\delta$ | A transition function, where |

$$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \text{finite subsets of } Q \times \Gamma^*$$

- Consider the various parts of $\delta$:

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

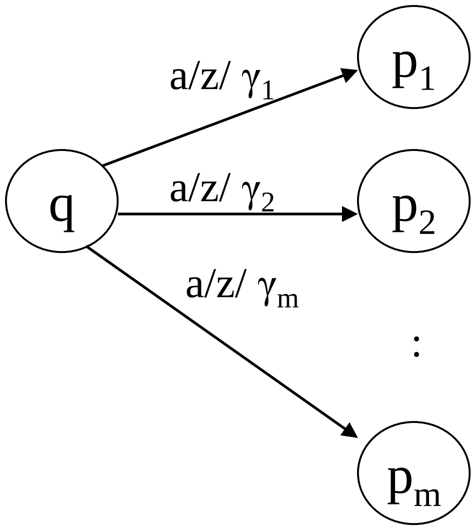
  - Q on the LHS means that at each step in a computation, a PDA must consider its' current state.
  - $\Gamma$ on the LHS means that at each step in a computation, a PDA must consider the symbol on top of its' stack.
  - $\Sigma \cup \{\varepsilon\}$ on the LHS means that at each step in a computation, 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 will have several options.
  - Q on the RHS means that each option specifies a new state.
  - $\Gamma^*$ on the RHS means that each option specifies zero or more stack symbols that will replace the top stack symbol.

- **Two types of PDA transitions:**

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

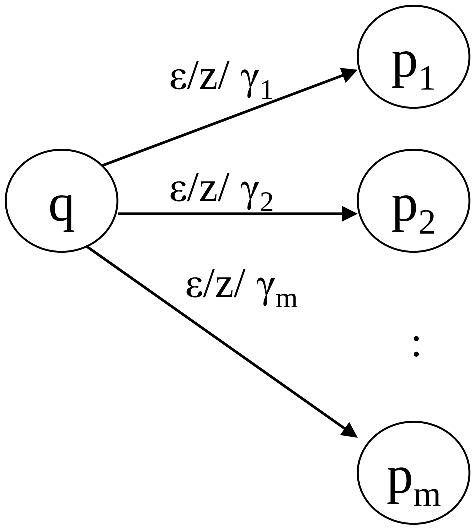
  - Current state is q
  - Current input symbol is a
  - Symbol currently on top of the stack z
  - Move to state $p_i$ from q
  - Replace z with $\gamma_i$ on the stack (leftmost symbol on top)
  - Move the input head to the next input symbol



- **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 $\gamma_i$ on the stack (leftmost symbol on top)
  - **No input symbol is read**



- **Example PDA #1:** (balanced parentheses)

  () (()) (())() ()((()))(())() $\varepsilon$

  Question: How could we accept the language with a stack-based Java program?

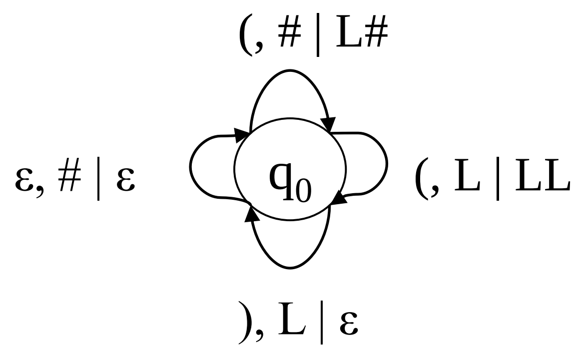
  $M = (\{q_1\}, \{ ( , ) \}, \{L, \#\}, \delta, q_1, \#, \varnothing)$

| | | | |
|---|---|---|---|
| $\delta$: | (1) | $\delta(q_1, (, \#) = \{(q_1, L\#)\}$ | // push a left paren |
| | (2) | $\delta(q_1, ), \#) = \varnothing$ | // too many right parens, reject |
| | (3) | $\delta(q_1, (, L) = \{(q_1, LL)\}$ | // push a left paren |
| | (4) | $\delta(q_1, ), L) = \{(q_1, \varepsilon)\}$ | // match a left and right paren |
| | (5) | $\delta(q_1, \varepsilon, \#) = \{(q_1, \varepsilon)\}$ | // empty the stack; accept |
| | (6) | $\delta(q_1, \varepsilon, L) = \varnothing$ | // too many left parens |

- **Goal:** (acceptance)
  - Terminate in a state
  - 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.

- Transition Diagram:




- Note that the above is not particularly illuminating.

- This is true for just about all PDAs, and consequently we don't typically draw the transition diagram.

* More generally, states are not particularly important in a PDA.

- **Example Computation:**

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

  $\delta$:

  | | | |
  |---|---|---|
  | (1) | $\delta(q_1, (, \#) = \{(q_1, L\#)\}$ | // push a left paren |
  | (2) | $\delta(q_1, ), \#) = \emptyset$ | // too many right parens, reject |
  | (3) | $\delta(q_1, (, L) = \{(q_1, LL)\}$ | // push a left paren |
  | (4) | $\delta(q_1, ), L) = \{(q_1, \varepsilon)\}$ | // match a left and right paren |
  | (5) | $\delta(q_1, \varepsilon, \#) = \{(q_1, \varepsilon)\}$ | // empty the stack; accept |
  | (6) | $\delta(q_1, \varepsilon, L) = \emptyset$ | // too many left parens |

  | Current Input | Stack | Rules Applicable | Rule Applied | |
  |---|---|---|---|---|
  | (()) | # | (1), (5) | (1) | --Why not 5? |
  | ()) | L# | (3), (6) | (3) | |
  | )) | LL# | (4), (6) | (4) | |
  | ) | L# | (4), (6) | (4) | |
  | ε | # | (5) | (5) | |
  | ε | ε | - | - | |

- Note that from this point forward, rules such as (2) and (6) will not be listed or referenced in any computations.

- **Example PDA #2:** For the language $\{x \mid x = wcw^r$ and $w$ in $\{0,1\}^*\}$

01c10 1101c1011 0010c0100 c

Question: How could we accept the language with a stack-based Java program?

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

$\delta$:

| | | | |
|---|---|---|---|
| (1) | $\delta(q_1, 0, R) = \{(q_1, BR)\}$ | (9) | $\delta(q_1, 1, R) = \{(q_1, GR)\}$ |
| (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) | $\delta(q_1, c, R) = \{(q_2, R)\}$ | | |
| (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) | $\delta(q_2, \varepsilon, R) = \{(q_2, \varepsilon)\}$ | | |

- **Notes:**
  - Rule #8 is used to pop the final stack symbol off at the end of a computation.

- **Example Computation:**

| | | | |
|---|---|---|---|
| (1) | $\delta(q_1, 0, R) = \{(q_1, BR)\}$ | (9) | $\delta(q_1, 1, R) = \{(q_1, GR)\}$ |
| (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) | $\delta(q_1, c, R) = \{(q_2, R)\}$ | | |
| (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) | $\delta(q_2, \varepsilon, R) = \{(q_2, \varepsilon)\}$ | | |

| State | Input | Stack | Rules Applicable | Rule Applied |
|---|---|---|---|---|
| $q_1$ | 01c10 | R | (1) | (1) |
| $q_1$ | 1c10 | BR | (10) | (10) |
| $q_1$ | c10 | GBR | (6) | (6) |
| $q_2$ | 10 | GBR | (12) | (12) |
| $q_2$ | 0 | BR | (7) | (7) |
| $q_2$ | $\varepsilon$ | R | (8) | (8) |
| $q_2$ | $\varepsilon$ | $\varepsilon$ | - | - |

- **Example Computation:**

| | | | |
|---|---|---|---|
| (1) | $\delta(q_1, 0, R) = \{(q_1, BR)\}$ | (9) | $\delta(q_1, 1, R) = \{(q_1, GR)\}$ |
| (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) | $\delta(q_1, c, R) = \{(q_2, R)\}$ | | |
| (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) | $\delta(q_2, \varepsilon, R) = \{(q_2, \varepsilon)\}$ | | |

| State | Input | Stack | Rules Applicable | Rule Applied |
|---|---|---|---|---|
| $q_1$ | 1c1 | R | (9) | (9) |
| $q_1$ | c1 | GR | (6) | (6) |
| $q_2$ | 1 | GR | (12) | (12) |
| $q_2$ | $\varepsilon$ | R | (8) | (8) |
| $q_2$ | $\varepsilon$ | $\varepsilon$ | - | - |

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

- **Example PDA #3:** For the language {x | x = $ww^r$ and w in {0,1}*}

  Without the "c" in the middle, switching from LHS processing to RHS processing is a challenge, because the PDA only "inputs" one symbol at a time.

  Assume the string is in the above language, where is the middle?

  0….
  01…
  010…
  0101…
  01011…
  010110…
  0101100…

  Two adjacent, identical symbols might indicate the middle position, but not necessarily.

  The best the PDA can do, is "guess" when it is in the middle.

- **Example PDA #3:** For the language $\{x \mid x = ww^r \text{ and } w \text{ in } \{0,1\}^*\}$

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

  $\delta$:

  (1) $\delta(q_1, 0, R) = \{(q_1, BR)\}$

  (2) $\delta(q_1, 1, R) = \{(q_1, GR)\}$

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

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

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

- **Example Computation:**

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

| State | Input | Stack | Rules Applicable | Rule Applied |
|---|---|---|---|---|
| $q_1$ | 000000 | R | (1), (9) | (1) |
| $q_1$ | 00000 | BR | (3), both options | (3), option #1 |
| $q_1$ | 0000 | BBR | (3), both options | (3), option #1 |
| $q_1$ | 000 | BBBR | (3), both options | (3), option #2 |
| $q_2$ | 00 | BBR | (7) | (7) |
| $q_2$ | 0 | BR | (7) | (7) |
| $q_2$ | $\varepsilon$ | R | (10) | (10) |
| $q_2$ | $\varepsilon$ | $\varepsilon$ | - | - |

- **Questions:**
  – What is rule #10 used for?
  – What is rule #9 used for?
  – Why do rules #3 and #6 have options?
  – Why don’t rules #4 and #5 have similar options?

- **Example Computation:**

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

| State | Input | Stack | Rules Applicable | Rule Applied |
|---|---|---|---|---|
| $q_1$ | 010010 | R | (1), (9) | (1) |
| $q_1$ | 10010 | BR | (5) | (5) |
| $q_1$ | 0010 | GBR | (4) | (4) |
| $q_1$ | 010 | BGBR | (3), both options | (3), option #2 |
| $q_2$ | 10 | GBR | (8) | (8) |
| $q_2$ | 0 | BR | (7) | (7) |
| $q_2$ | $\varepsilon$ | R | (10) | (10) |
| $q_2$ | $\varepsilon$ | $\varepsilon$ | - | - |

- **Exercises:**
  - 0011001100
  - 011110
  - 0111

**Exercises:**

- Develop PDAs for any of the regular or context-free languages that we have discussed.
- Note that for regular languages an NFA that simply "ignores" it's stack will work.
- For languages which are context-free but not regular, first try to envision a Java (or other high-level language) program that uses a stack to accept the language, and then convert it to a PDA.
- For example, for the set of all strings of the form $a^i b^j c^k$, such that either $i \neq j$ or $j \neq k$. Or the set of all strings not of the form ww.

# 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 $(q, w, \gamma)$, where q is in Q, w is in $\Sigma^*$ and $\gamma$ is in $\Gamma^*$.
  - q is the current state
  - w is the unused input
  - $\gamma$ is the current stack contents

- **Example:** (for PDA #3)

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

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

- Intuitively, if I and J are instantaneous descriptions, then I |— J means that J follows from I by one transition.

- **Formally:** Let a be in Σ U {ε}, w be in Σ*, z be in Γ, and α and β both be in Γ*. Then:

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

if δ(q, a, z) contains (p, β).

- **Examples:** (PDA #3)

$(q_1, 111, GBR) \vdash (q_1, 11, GGBR)$ (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)

- A computation by a PDA can be expressed using this notation (PDA #3):

| | | |
|---|---|---|
| $(q_1, 010010, R)$ | $\vdash (q_1, 10010, BR)$ | (1) |
| | $\vdash (q_1, 0010, GBR)$ | (5) |
| | $\vdash (q_1, 010, BGBR)$ | (4) |
| | $\vdash (q_2, 10, GBR)$ | (3), option #2 |
| | $\vdash (q_2, 0, BR)$ | (8) |
| | $\vdash (q_2, \varepsilon, R)$ | (7) |
| | $\vdash (q_2, \varepsilon, \varepsilon)$ | (10) |

$(q_1, \varepsilon, R) \vdash (q_2, \varepsilon, \varepsilon)$ (9)

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

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

- **Alternatively:**
  - I |—* I for each instantaneous description I
  - If I |—* J and J |— K then I |—* K

- **Examples:** (PDA #3)

  $(q_1, 010010, R) \vdash^* (q_2, 10, GBR)$

  $(q_1, 010010, R) \vdash^* (q_2, \varepsilon, \varepsilon)$

  $(q_1, 111, GBR) \vdash^* (q_1, \varepsilon, GGGGBR)$

  $(q_1, 01, GR) \vdash^* (q_1, 1, BGR)$

  $(q_1, 101, GBR) \vdash^* (q_1, 101, GBR)$

- **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

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

- **Questions:**
  - Does the book define string acceptance by empty stack, final state, both, or neither?
  - As an exercise, convert the preceding PDAs to other PDAs with different acceptence criteria.

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

- **Theorem:** Let L be a language. Then there exists 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.

- **Notes:**
  - Similar lemmas and theorems could be stated for PDAs that accept by both final state and empty stack.
  - Part of the lesson here is that one can define “acceptance” in many different ways, e.g., a string is accepted by a DFA if you simply pass through an accepting state, or if you pass through an accepting state exactly twice.

# The Relationship Between PDAs and CFLs

- **Definition:** Let G = (V, T, P, S) be a CFG. 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 Greibach Normal Form (GNF).

- **Example:**

$S \rightarrow aAB \mid bB$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid c$

- **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, where L = L(G), 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 T, A in V and γ in V*, if A –> aγ is in P then δ(q, a, A) will contain (q, γ)

  Stated another way:
  δ(q, a, A) = {(q, γ) | A –> aγ is in P}, for all a in T and A in V

- Huh?

- As we will see, 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:

  $Q = \{q\}$
  $\Sigma = T = \{a\}$
  $\Gamma = V = \{S\}$
  $z = S$

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

- **Question:** Is that all? Is $\delta$ complete? Recall that $\delta$: $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow$ finite subsets of $Q \times \Gamma^*$

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

(1) $S \rightarrow aA$
(2) $S \rightarrow aB$
(3) $A \rightarrow aA$
(4) $A \rightarrow aB$
(5) $B \rightarrow bB$
(6) $B \rightarrow b$

G is in GNF
$L(G) = a^+b^+$

Construct M as:
$Q = \{q\}$
$\Sigma = T = \{a, b\}$
$\Gamma = V = \{S, A, B\}$
$z = S$

$S \rightarrow a\gamma$ How many productions are there of this form?

(1) $\delta(q, a, S) = ?$
(2) $\delta(q, a, A) = ?$
(3) $\delta(q, a, B) = ?$
(4) $\delta(q, b, S) = ?$
(5) $\delta(q, b, A) = ?$
(6) $\delta(q, b, B) = ?$
(7) $\delta(q, \varepsilon, S) = ?$
(8) $\delta(q, \varepsilon, A) = ?$
(9) $\delta(q, \varepsilon, B) = ?$

Why 9? Recall $\delta$: $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow$ finite subsets of $Q \times \Gamma^*$

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

(1) $S \rightarrow aA$
(2) $S \rightarrow aB$
(3) $A \rightarrow aA$
(4) $A \rightarrow aB$
(5) $B \rightarrow bB$
(6) $B \rightarrow b$

G is in GNF
$L(G) = a^+b^+$

Construct M as:
$Q = \{q\}$
$\Sigma = T = \{a, b\}$
$\Gamma = V = \{S, A, B\}$
$z = S$

$S \rightarrow a\gamma$ How many productions are there of this form?

(1) $\delta(q, a, S) = \{(q, A), (q, B)\}$ From productions #1 and 2, S->aA, S->aB
(2) $\delta(q, a, A) = ?$
(3) $\delta(q, a, B) = ?$
(4) $\delta(q, b, S) = ?$
(5) $\delta(q, b, A) = ?$
(6) $\delta(q, b, B) = ?$
(7) $\delta(q, \varepsilon, S) = ?$
(8) $\delta(q, \varepsilon, A) = ?$
(9) $\delta(q, \varepsilon, B) = ?$ Why 9? Recall $\delta$: $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow$ finite subsets of $Q \times \Gamma^*$

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

(1) S –> aA
(2) S –> aB
(3) A –> aA
(4) A –> aB
(5) B –> bB
(6) B –> b

G is in GNF

$L(G) = a^+b^+$

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 | |
| (t9) δ(q, ε, B) = Ø | Recall δ: Q x (Σ U {ε}) x Γ –> finite subsets of Q x Γ* |

- 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:

$$\Rightarrow t_1t_2\ldots t_i\, A_1A_2\ldots A_m$$

terminals non-terminals

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

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

$$\Rightarrow t_1t_2\ldots t_i\, t_{i+1}\, \alpha A_2\ldots A_m$$

- 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.

- After the $i^{th}$ step of the computation of the PDA, $t_1t_2\ldots t_i$ are the symbols that have already been read by the PDA and $A_1A_2\ldots A_m$ 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 | (p1) |
| | => aaA | (p3) |
| | => aaaA | (p3) |
| | => aaaaB | (p4) |
| | => aaaabB | (p5) |
| | => aaaabb | (p6) |

- The corresponding computation of the PDA:

- (q, aaaabb, S) $\vdash$ ?

(p1) S –> aA
(p2) S –> aB
(p3) A –> aA
(p4) A –> aB
(p5) B –> bB
(p6) B –> b

| | | |
|---|---|---|
| (t1) | $\delta(q, a, S) = \{(q, A), (q, B)\}$ | productions p1 and p2 |
| (t2) | $\delta(q, a, A) = \{(q, A), (q, B)\}$ | productions p3 and p4 |
| (t3) | $\delta(q, a, B) = \emptyset$ | |
| (t4) | $\delta(q, b, S) = \emptyset$ | |
| (t5) | $\delta(q, b, A) = \emptyset$ | |
| (t6) | $\delta(q, b, B) = \{(q, B), (q, \varepsilon)\}$ | productions p5 and p6 |
| (t7) | $\delta(q, \varepsilon, S) = \emptyset$ | |
| (t8) | $\delta(q, \varepsilon, A) = \emptyset$ | |
| (t9) | $\delta(q, \varepsilon, B) = \emptyset$ | |

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

| | | |
|---|---|---|
| S | => aA | (p1) |
| | => aaA | (p3) |
| | => aaaA | (p3) |
| | => aaaaB | (p4) |
| | => aaaabB | (p5) |
| | => aaaabb | (p6) |

- The corresponding computation of the PDA:

- (q, aaaabb, S) |— (q, aaabb, A) (t1)/1
  |— (q, aabb, A) (t2)/1
  |— (q, abb, A) (t2)/1
  |— (q, bb, B) (t2)/2
  |— (q, b, B) (t6)/1
  |— (q, ε, ε) (t6)/2

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

(p1) S –> aA
(p2) S –> aB
(p3) A –> aA
(p4) A –> aB
(p5) B –> bB
(p6) B –> b

| | | |
|---|---|---|
| (t1) | $\delta(q, a, S) = \{(q, A), (q, B)\}$ | productions p1 and p2 |
| (t2) | $\delta(q, a, A) = \{(q, A), (q, B)\}$ | productions p3 and p4 |
| (t3) | $\delta(q, a, B) = \emptyset$ | |
| (t4) | $\delta(q, b, S) = \emptyset$ | |
| (t5) | $\delta(q, b, A) = \emptyset$ | |
| (t6) | $\delta(q, b, B) = \{(q, B), (q, \varepsilon)\}$ | productions p5 and p6 |
| (t7) | $\delta(q, \varepsilon, S) = \emptyset$ | |
| (t8) | $\delta(q, \varepsilon, A) = \emptyset$ | |
| (t9) | $\delta(q, \varepsilon, B) = \emptyset$ | |

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

$$
\begin{aligned}
(q, aabb, S) &\vdash (q, abb, A) && (t1)/1 \\
&\vdash (q, bb, B) && (t2)/2 \\
&\vdash (q, b, B) && (t6)/1 \\
&\vdash (q, \varepsilon, \varepsilon) && (t6)/2
\end{aligned}
$$

- The corresponding derivation using the grammar:

S => ?

(p1) S –> aA
(p2) S –> aB
(p3) A –> aA
(p4) A –> aB
(p5) B –> bB
(p6) B –> b

| | | |
|---|---|---|
| (t1) | $\delta(q, a, S) = \{(q, A), (q, B)\}$ | productions p1 and p2 |
| (t2) | $\delta(q, a, A) = \{(q, A), (q, B)\}$ | productions p3 and p4 |
| (t3) | $\delta(q, a, B) = \varnothing$ | |
| (t4) | $\delta(q, b, S) = \varnothing$ | |
| (t5) | $\delta(q, b, A) = \varnothing$ | |
| (t6) | $\delta(q, b, B) = \{(q, B), (q, \varepsilon)\}$ | productions p5 and p6 |
| (t7) | $\delta(q, \varepsilon, S) = \varnothing$ | |
| (t8) | $\delta(q, \varepsilon, A) = \varnothing$ | |
| (t9) | $\delta(q, \varepsilon, B) = \varnothing$ | |

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

$(q, aabb, S) \vdash (q, abb, A)$ (t1)/1
$\vdash (q, bb, B)$ (t2)/2
$\vdash (q, b, B)$ (t6)/1
$\vdash (q, \varepsilon, \varepsilon)$ (t6)/2

- The corresponding derivation using the grammar:

S => aA (p1)
=> aaB (p4)
=> aabB (p5)
=> aabb (p6)

(p1) S –> aA
(p2) S –> aB
(p3) A –> aA
(p4) A –> aB
(p5) B –> bB
(p6) B –> b

(t1) $\delta(q, a, S) = \{(q, A), (q, B)\}$ productions p1 and p2
(t2) $\delta(q, a, A) = \{(q, A), (q, B)\}$ productions p3 and p4
(t3) $\delta(q, a, B) = \emptyset$
(t4) $\delta(q, b, S) = \emptyset$
(t5) $\delta(q, b, A) = \emptyset$
(t6) $\delta(q, b, B) = \{(q, B), (q, \varepsilon)\}$ productions p5 and p6
(t7) $\delta(q, \varepsilon, S) = \emptyset$
(t8) $\delta(q, \varepsilon, A) = \emptyset$
(t9) $\delta(q, \varepsilon, B) = \emptyset$

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

| | | |
|---|---|---|
| (1) | S –> aABC | |
| (2) | A –> a | G is in GNF |
| (3) | B –> b | |
| (4) | C –> cAB | |
| (5) | C –> cC | |

Construct M as:

$Q = \{q\}$
$\Sigma = T = \{a, b, c\}$
$\Gamma = V = \{S, A, B, C\}$
$z = S$

| | | | | | |
|---|---|---|---|---|---|
| (1) | $\delta(q, a, S) = \{(q, ABC)\}$ | S->aABC | (9) | $\delta(q, c, S) = \emptyset$ | |
| (2) | $\delta(q, a, A) = \{(q, \varepsilon)\}$ | A->a | (10) | $\delta(q, c, A) = \emptyset$ | |
| (3) | $\delta(q, a, B) = \emptyset$ | | (11) | $\delta(q, c, B) = \emptyset$ | |
| (4) | $\delta(q, a, C) = \emptyset$ | | (12) | $\delta(q, c, C) = \{(q, AB), (q, C))$ | C->cAB\|cC |
| (5) | $\delta(q, b, S) = \emptyset$ | | (13) | $\delta(q, \varepsilon, S) = \emptyset$ | |
| (6) | $\delta(q, b, A) = \emptyset$ | | (14) | $\delta(q, \varepsilon, A) = \emptyset$ | |
| (7) | $\delta(q, b, B) = \{(q, \varepsilon)\}$ | B->b | (15) | $\delta(q, \varepsilon, B) = \emptyset$ | |
| (8) | $\delta(q, b, C) = \emptyset$ | | (16) | $\delta(q, \varepsilon, C) = \emptyset$ | |

- **Notes:**
  - Recall that the grammar G was required to be in GNF before the construction could be applied.
  - As a result, it was assumed that ε was not in the context-free language L.

- **Suppose ε is in L:**

1) First, let L’ = L – {ε}

By an earlier theorem, if L is a CFL, then L’ = L – {ε} is a CFL.

By another 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 $L = \{\varepsilon, b, ab, aab, aaab, \ldots\}$ Then $L' = \{b, ab, aab, aaab, \ldots\}$

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

(1) S –> aS
(2) S –> b

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

$Q = \{q\}$
$\Sigma = T = \{a, b\}$
$\Gamma = V = \{S\}$
$z = S$

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

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

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

3) Instead, add a new *start* state q’ with transitions:

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

where q is the start state of the machine from the initial construction.

- **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 $L_E(M) = L(G)$ . -- Note that we did not prove this.
- **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.

- A (proposed) PDA for $\{0^i1^j2^k \mid i \neq j \text{ or } j \neq k\}$

- For simplicity assume that i,j,k>=1

- There are four cases i>j, i<j, j>k, j<k

- The PDA uses epsilon transitions to guess which case holds

| | | |
|---|---|---|
| (1) | $\delta(q_0, \varepsilon, \#) = \{(q_1, \#), (q_2, \#), (q_3, \#), (q_4, \#)\}$ | // Guess which of the four cases applies |
| (4) | $\delta(q_1, 0, \#) = \{(q_1, 0\#)\}$ | // This begins case 1, start by pushing all the 0's |
| (5) | $\delta(q_1, 0, 0) = \{(q_1, 00)\}$ | |
| (6) | $\delta(q_1, 1, 0) = \{(q_5, \varepsilon)\}$ | // Match the 1's on input with the 0's on the stack |
| (7) | $\delta(q_5, 1, 0) = \{(q_5, \varepsilon)\}$ | |
| (8) | $\delta(q_5, 2, 0) = \{(q_6, 0)\}$ | // 1's run out first, so look for a 2 and eat them up |
| (9) | $\delta(q_6, 2, 0) = \{(q_6, 0)\}$ | |
| (10) | $\delta(q_6, \varepsilon, 0) = \{(q_7, \varepsilon)\}$ | // Once 2's run out, empty the stack, and accept |
| (11) | $\delta(q_7, \varepsilon, 0) = \{(q_7, \varepsilon)\}$ | |
| | : | // Cases 2-4 are similar |