Pushdown Automata

Reading: Chapter 6
Pushdown Automata (PDA)

- **Informally:**
  - A PDA is an NFA-ε with an infinite 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:**

\[ \{0^n1^n \mid 0 \leq n \} \quad \text{Is not regular} \]

\[ \{0^n1^n \mid 0 \leq n \leq k, \text{for some fixed } k \} \quad \text{Is regular, for any fixed } k. \]

• **For } k=3:\]

\[ L = \{ \varepsilon, 01, 0011, 000111 \} \]
• In a DFA, each state remembers a finite amount of information.

• To get \( \{0^n1^n \mid n \geq 0\} \) with a DFA would require an infinite number of states using the preceding technique.

• An infinite stack solves the problem for \( \{0^n1^n \mid 0 \leq n\} \) 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 \( \{0^n1^m0^{n+m} \mid n,m \geq 0\} \)
Formal Definition of a PDA

- A pushdown automaton (PDA) is a seven-tuple:

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

  - \( Q \) – A finite set of states
  - \( \Sigma \) – A finite input alphabet
  - \( \Gamma \) – A finite 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

  \[ \delta: Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \text{finite subsets of } Q \times \Gamma^* \]
Consider the various parts of $\delta$:

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

- $Q$ 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)

\[
() \quad (()) \quad ()() \quad ()((())(())() \quad \varepsilon
\]

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

\[
M = (\{q_1\}, \{\text{"\(\)"}, \text{"\(\)"}\}, \{L, \#\}, \delta, q_1, \#, \emptyset)
\]

\[
\delta: \quad \begin{align*}
(1) & \quad \delta(q_1, (, \#) = \{(q_1, L\#)\} \\
(2) & \quad \delta(q_1, ), \#) = \emptyset \\
(3) & \quad \delta(q_1, (, L) = \{(q_1, LL)\} \\
(4) & \quad \delta(q_1, ), L) = \{(q_1, \varepsilon)\} \\
(5) & \quad \delta(q_1, \varepsilon, \#) = \{(q_1, \varepsilon)\} \\
(6) & \quad \delta(q_1, \varepsilon, L) = \emptyset
\end{align*}
\]

• **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\}, \{\text{"\(\),\"}\}, \{L, \#\}, \delta, q_1, \#, \emptyset) \]

\[ \delta: \]

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

(2) \[ \delta(q_1, ), \#) = \emptyset \]

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

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

(5) \[ \delta(q_1, \varepsilon, \#) = \{(q_1, \varepsilon)\} \]

(6) \[ \delta(q_1, \varepsilon, L) = \emptyset \]

<table>
<thead>
<tr>
<th>Current Input</th>
<th>Stack</th>
<th>Rules Applicable</th>
<th>Rule Applied</th>
</tr>
</thead>
<tbody>
<tr>
<td>()</td>
<td>#</td>
<td>(1), (5)</td>
<td>(1) --Why not 5?</td>
</tr>
<tr>
<td>()</td>
<td>L#</td>
<td>(3), (6)</td>
<td>(3)</td>
</tr>
<tr>
<td>))</td>
<td>LL#</td>
<td>(4), (6)</td>
<td>(4)</td>
</tr>
<tr>
<td>)</td>
<td>L#</td>
<td>(4), (6)</td>
<td>(4)</td>
</tr>
<tr>
<td>\varepsilon</td>
<td>#</td>
<td>(5)</td>
<td>(5)</td>
</tr>
<tr>
<td>\varepsilon</td>
<td>\varepsilon</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

• 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 \( \{ x \mid x = wcw^r \text{ and } w \in \{0,1\}^* \} \)

\[
\begin{array}{cccccc}
01c10 & 1101c1011 & 0010c0100 & c
\end{array}
\]

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:

\[
\begin{align*}
(1) \quad \delta(q_1, 0, R) &= \{(q_1, BR)\} & (9) \quad \delta(q_1, 1, R) &= \{(q_1, GR)\} \\
(2) \quad \delta(q_1, 0, B) &= \{(q_1, BB)\} & (10) \quad \delta(q_1, 1, B) &= \{(q_1, GB)\} \\
(3) \quad \delta(q_1, 0, G) &= \{(q_1, BG)\} & (11) \quad \delta(q_1, 1, G) &= \{(q_1, GG)\} \\
(4) \quad \delta(q_1, c, R) &= \{(q_2, R)\} \\
(5) \quad \delta(q_1, c, B) &= \{(q_2, B)\} \\
(6) \quad \delta(q_1, c, G) &= \{(q_2, G)\} \\
(7) \quad \delta(q_2, 0, B) &= \{(q_2, \varepsilon)\} & (12) \quad \delta(q_2, 1, G) &= \{(q_2, \varepsilon)\} \\
(8) \quad \delta(q_2, \varepsilon, R) &= \{(q_2, \varepsilon)\} \\
(8) \quad \delta(q_2, \varepsilon, R) &= \{(q_2, \varepsilon)\}
\end{align*}
\]

<table>
<thead>
<tr>
<th>State</th>
<th>Input</th>
<th>Stack</th>
<th>Rules Applicable</th>
<th>Rule Applied</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₁</td>
<td>01c10</td>
<td>R</td>
<td>(1)</td>
<td>(1)</td>
</tr>
<tr>
<td>q₁</td>
<td>1c10</td>
<td>BR</td>
<td>(10)</td>
<td>(10)</td>
</tr>
<tr>
<td>q₁</td>
<td>c10</td>
<td>GBR</td>
<td>(6)</td>
<td>(6)</td>
</tr>
<tr>
<td>q₂</td>
<td>10</td>
<td>GBR</td>
<td>(12)</td>
<td>(12)</td>
</tr>
<tr>
<td>q₂</td>
<td>0</td>
<td>BR</td>
<td>(7)</td>
<td>(7)</td>
</tr>
<tr>
<td>q₂</td>
<td>\varepsilon</td>
<td>R</td>
<td>(8)</td>
<td>(8)</td>
</tr>
<tr>
<td>q₂</td>
<td>\varepsilon</td>
<td>\varepsilon</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
• **Example Computation:**

1. \( \delta(q_1, 0, R) = \{(q_1, BR)\} \)
2. \( \delta(q_1, 0, B) = \{(q_1, BB)\} \)
3. \( \delta(q_1, 0, G) = \{(q_1, BG)\} \)
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. \( \delta(q_2, 0, B) = \{(q_2, \varepsilon)\} \)
8. \( \delta(q_2, \varepsilon, R) = \{(q_2, \varepsilon)\} \)
9. \( \delta(q_1, 1, R) = \{(q_1, GR)\} \)
10. \( \delta(q_1, 1, B) = \{(q_1, GB)\} \)
11. \( \delta(q_1, 1, G) = \{(q_1, GG)\} \)
12. \( \delta(q_2, 1, G) = \{(q_2, \varepsilon)\} \)

<table>
<thead>
<tr>
<th>State</th>
<th>Input</th>
<th>Stack</th>
<th>Rules Applicable</th>
<th>Rule Applied</th>
</tr>
</thead>
<tbody>
<tr>
<td>(q_1)</td>
<td>1c1</td>
<td>R</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>(q_1)</td>
<td>c1</td>
<td>GR</td>
<td>(6)</td>
<td>(6)</td>
</tr>
<tr>
<td>(q_2)</td>
<td>1</td>
<td>GR</td>
<td>(12)</td>
<td>(12)</td>
</tr>
<tr>
<td>(q_2)</td>
<td>(\varepsilon)</td>
<td>R</td>
<td>(8)</td>
<td>(8)</td>
</tr>
<tr>
<td>(q_2)</td>
<td>(\varepsilon)</td>
<td>(\varepsilon)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

• **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 \( \{ x \mid x = w w^r \text{ and } w \in \{0,1\}^* \} \)

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

\[
\delta:
\]

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

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

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

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

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

(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) \( \delta(q_1, 0, R) = \{(q_1, BR)\} \)
(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) \( \delta(q_2, \varepsilon, R) = \{(q_2, \varepsilon)\} \)

<table>
<thead>
<tr>
<th>State</th>
<th>Input</th>
<th>Stack</th>
<th>Rules Applicable</th>
<th>Rule Applied</th>
</tr>
</thead>
<tbody>
<tr>
<td>q_1</td>
<td>000000</td>
<td>R</td>
<td>(1), (9)</td>
<td>(1)</td>
</tr>
<tr>
<td>q_1</td>
<td>0000</td>
<td>BR</td>
<td>(3), both options</td>
<td>(3), option #1</td>
</tr>
<tr>
<td>q_1</td>
<td>000</td>
<td>BBR</td>
<td>(3), both options</td>
<td>(3), option #1</td>
</tr>
<tr>
<td>q_1</td>
<td>00</td>
<td>BBBR</td>
<td>(3), both options</td>
<td>(3), option #2</td>
</tr>
<tr>
<td>q_2</td>
<td>0</td>
<td>BBR</td>
<td>(7)</td>
<td>(7)</td>
</tr>
<tr>
<td>q_2</td>
<td>\varepsilon</td>
<td>R</td>
<td>(10)</td>
<td>(10)</td>
</tr>
<tr>
<td>q_2</td>
<td>\varepsilon</td>
<td>\varepsilon</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

• **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) \( \delta(q_1, 0, R) = \{ (q_1, BR) \} \)  
(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) \( \delta(q_2, \varepsilon, R) = \{ (q_2, \varepsilon) \} \)  


<table>
<thead>
<tr>
<th>State</th>
<th>Input</th>
<th>Stack</th>
<th>Rules Applicable</th>
<th>Rule Applied</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_1 )</td>
<td>0100</td>
<td>R</td>
<td>(1), (9)</td>
<td>(1)</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>100</td>
<td>BR</td>
<td>(5)</td>
<td>(5)</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>000</td>
<td>GBR</td>
<td>(4)</td>
<td>(4)</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>01</td>
<td>BGBR</td>
<td>(3), both options</td>
<td>(3), option #2</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>10</td>
<td>GBR</td>
<td>(8)</td>
<td>(8)</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>0</td>
<td>BR</td>
<td>(7)</td>
<td>(7)</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( \varepsilon )</td>
<td>R</td>
<td>(10)</td>
<td>(10)</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

• **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 its 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)

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

  $(q_1, 111, GBR)$ \quad $(q_2, 11, BR)$

  $(q_1, 000, GR)$ \quad $(q_2, 00, R)$
• 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 $\Sigma^*$, $z$ be in $\Gamma$, and $\alpha$ and $\beta$ both be in $\Gamma^*$. Then:

$$(q, aw, z\alpha) \xrightarrow{M} (p, w, \beta\alpha)$$

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

• Intuitively, if $I$ and $J$ are instantaneous descriptions, then $I \xrightarrow{\quad} J$ means that $J$ follows from $I$ by one transition.
• **Examples:** (PDA #3)

\[(q_1, 111, GBR) \longrightarrow (q_1, 11, GGBR)\]  
(6) option #1, with \(a=1\), \(z=G\), \(\beta=GG\), \(w=11\), and \(\alpha=BR\)

\[(q_1, 111, GBR) \longrightarrow (q_2, 11, BR)\]  
(6) option #2, with \(a=1\), \(z=G\), \(\beta=\varepsilon\), \(w=11\), and \(\alpha=BR\)

\[(q_1, 000, GR) \longrightarrow (q_2, 00, R)\]  
Is not true, For any \(a\), \(z\), \(\beta\), \(w\) and \(\alpha\)

• **Examples:** (PDA #1)

\[(q_1, (0)), L\#) \longrightarrow (q_1, ()), LL\#)\]  
(3)
A computation by a PDA can be expressed using this notation (PDA #3):

\[
\begin{align*}
(q_1, 010010, \text{R}) & \rightarrow (q_1, 10010, \text{BR}) & (1) \\
& \rightarrow (q_1, 0010, \text{GBR}) & (5) \\
& \rightarrow (q_1, 010, \text{BGBR}) & (4) \\
& \rightarrow (q_2, 10, \text{GBR}) & (3), \text{ option } #2 \\
& \rightarrow (q_2, 0, \text{BR}) & (8) \\
& \rightarrow (q_2, \varepsilon, \text{R}) & (7) \\
& \rightarrow (q_2, \varepsilon, \varepsilon) & (10)
\end{align*}
\]

\[
(q_1, \varepsilon, \text{R}) \rightarrow (q_2, \varepsilon, \varepsilon) & \quad (9)
\]
• **Definition:** $\vdash *$ is the reflexive and transitive closure of $\vdash$.
  
  – $I \vdash * I$ for each instantaneous description $I$
  
  – If $I \vdash J$ and $J \vdash * K$ then $I \vdash * K$

• Intuitively, if $I$ and $J$ are instantaneous descriptions, then $I \vdash * J$ means that $J$ follows from $I$ by zero or more transitions.

• **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

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

• **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

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

• **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

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

• **Questions:**
  - How does the formal definition of a PDA differ from that given in the book?
  - 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 acceptance 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” however they want, 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.
• **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 $\alpha$ 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 $L - \{\varepsilon\}$ is a CFL.

• **Theorem:** Let $L$ be a CFL not containing $\{\varepsilon\}$. 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 \( \varepsilon \) is not in \( L \). The construction can be modified to include \( \varepsilon \) 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 \( M = (Q, \Sigma, \Gamma, \delta, q, z, \emptyset) \) where:

\[
Q = \{q\} \\
\Sigma = T \\
\Gamma = V \\
z = S
\]

\( \delta \): for all \( a \) in \( T \), \( A \) in \( V \) and \( \gamma \) in \( V^* \), if \( A \rightarrow a\gamma \) is in \( P \) then \( \delta(q, a, A) \) will contain \( (q, \gamma) \)

Stated another way:

\[
\delta(q, a, A) = \{(q, \gamma) \mid A \rightarrow a\gamma \text{ is in } P\}, \text{ for all } a \text{ in } T \text{ and } A \text{ in } V
\]

• Huh?

• As we will see, for a given string \( x \) in \( \Sigma^* \), \( M \) will attempt to simulate a leftmost derivation of \( x \) with \( G \).
• **Example #1:** Consider the following CFG in GNF.

\[
\begin{align*}
S & \rightarrow aS \\
S & \rightarrow a
\end{align*}
\]

G is in GNF

\[L(G) = a^+\]

Construct M as:

\[
\begin{align*}
Q &= \{q\} \\
\Sigma = T &= \{a\} \\
\Gamma = V &= \{S\} \\
z &= S
\end{align*}
\]

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

\[
\delta(q, \varepsilon, S) = \emptyset
\]

• **Question:** Is that all? Is \(\delta\) complete? Recall that \(\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \text{finite subsets of } Q \times \Gamma^*\)
Example #2: Consider the following CFG in GNF.

(1) $S \rightarrow aA$
(2) $S \rightarrow aB$
(3) $A \rightarrow aA$  \hspace{1cm} G is in GNF
(4) $A \rightarrow aB$  \hspace{1cm} $L(G) = a^+b^+$
(5) $B \rightarrow bB$
(6) $B \rightarrow b$

Construct $M$ as:

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

(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) $\delta(q, \epsilon, S) =$ ?
(8) $\delta(q, \epsilon, A) =$ ?
(9) $\delta(q, \epsilon, B) =$ ?

Recall $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow$ finite subsets of $Q \times \Gamma^*$
• **Example #2:** Consider the following CFG in GNF.

(1) \( S \rightarrow aA \)
(2) \( S \rightarrow aB \)
(3) \( A \rightarrow aA \) \hspace{1cm} \text{G is in GNF}
(4) \( A \rightarrow aB \) \hspace{1cm} \( L(G) = a^+b^+ \)
(5) \( B \rightarrow bB \)
(6) \( B \rightarrow b \)

Construct \( M \) as:

\[ Q = \{ q \} \]
\[ \Sigma = T = \{ a, b \} \]
\[ \Gamma = V = \{ S, A, B \} \]
\[ z = S \]

(1) \( \delta(q, a, S) = \{(q, A), (q, B)\} \) \hspace{1cm} \text{From productions #1 and 2, } S \rightarrow aA, S \rightarrow aB
(2) \( \delta(q, a, A) = \{(q, A), (q, B)\} \) \hspace{1cm} \text{From productions #3 and 4, } A \rightarrow aA, A \rightarrow aB
(3) \( \delta(q, a, B) = \emptyset \)
(4) \( \delta(q, b, S) = \emptyset \)
(5) \( \delta(q, b, A) = \emptyset \)
(6) \( \delta(q, b, B) = \{(q, B), (q, \varepsilon)\} \) \hspace{1cm} \text{From productions #5 and 6, } B \rightarrow bB, B \rightarrow b
(7) \( \delta(q, \varepsilon, S) = \emptyset \)
(8) \( \delta(q, \varepsilon, A) = \emptyset \)
(9) \( \delta(q, \varepsilon, B) = \emptyset \) \hspace{1cm} \text{Recall } \delta: Q \times (\Sigma \cup \{ \varepsilon \}) \times \Gamma \to \text{finite subsets of } Q \times \Gamma^*
• For a string \( w \) in \( L(G) \) the PDA \( M \) will simulate a leftmost derivation of \( w \).
  
  – If \( w \) is in \( L(G) \) then \((q, w, z_0) \rightharpoonup^* (q, \varepsilon, \varepsilon)\)
  
  – If \((q, w, z_0) \rightharpoonup^* (q, \varepsilon, \varepsilon)\) then \( w \) is in \( L(G) \)

• Consider generating a string using \( G \). Since \( G \) is in GNF, each sentential form in a \textit{leftmost} derivation has form:

\[
=> \ t_1 t_2 \ldots t_i A_1 A_2 \ldots A_m
\]

  terminals \hspace{1cm} non-terminals

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

\[
A_i \rightarrow t_{i+1} \alpha
\]

\[
=> t_1 t_2 \ldots t_i t_{i+1} \alpha A_2 \ldots A_m
\]

• 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, \( t_1 t_2 \ldots t_i \) are the symbols that have already been read by the PDA and \( A_1 A_2 \ldots A_m \) 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:

  \[ \Rightarrow t_1t_2\ldots t_i A_1A_2\ldots A_m \]

  would be:

  \[(q, t_{i+1}t_{i+2}\ldots t_n, A_1A_2\ldots A_m)\]
• **Example:** Using the grammar from example #2:

\[
\begin{align*}
S & \Rightarrow aA \quad (1) \\
& \Rightarrow aaA \quad (3) \\
& \Rightarrow aaaA \quad (3) \\
& \Rightarrow aaaaB \quad (4) \\
& \Rightarrow aaaabB \quad (5) \\
& \Rightarrow aaaabb \quad (6)
\end{align*}
\]

• The corresponding computation of the PDA:

• \((q, \text{aaaabb}, S) \rightarrow ?\)
• **Example:** Using the grammar from example #2:

\[
S \rightarrow aA \quad (1) \\
   \rightarrow aaA \quad (3) \\
   \rightarrow aaaA \quad (3) \\
   \rightarrow aaaaB \quad (4) \\
   \rightarrow aaaabB \quad (5) \\
   \rightarrow aaaabb \quad (6)
\]

• The corresponding computation of the PDA:

\[
(q, \text{aaaabb}, S) \mid\rightarrow (q, \text{aaabb}, A) \quad (1)/1 \\
   \mid\rightarrow (q, \text{aabb}, A) \quad (2)/1 \\
   \mid\rightarrow (q, \text{abb}, A) \quad (2)/1 \\
   \mid\rightarrow (q, \text{bb}, B) \quad (2)/2 \\
   \mid\rightarrow (q, \text{b}, B) \quad (6)/1 \\
   \mid\rightarrow (q, \varepsilon, \varepsilon) \quad (6)/2
\]

- 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) |— (q, abb, A) (1)/1
|— (q, bb, B) (2)/2
|— (q, b, B) (6)/1
|— (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) \( S \rightarrow aABC \)
(2) \( A \rightarrow a \) \( G \) is in GNF
(3) \( B \rightarrow b \)
(4) \( C \rightarrow cAB \)
(5) \( C \rightarrow cC \)

Construct \( M \) as:

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

(1) \( \delta(q, a, S) = \{(q, ABC)\} \) \( S \rightarrow aABC \)
(2) \( \delta(q, a, A) = \{(q, \varepsilon)\} \) \( A \rightarrow a \)
(3) \( \delta(q, a, B) = \emptyset \)
(4) \( \delta(q, a, C) = \emptyset \)
(5) \( \delta(q, b, S) = \emptyset \)
(6) \( \delta(q, b, A) = \emptyset \)
(7) \( \delta(q, b, B) = \{(q, \varepsilon)\} \) \( B \rightarrow b \)
(8) \( \delta(q, b, C) = \emptyset \)
(9) \( \delta(q, c, S) = \emptyset \)
(10) \( \delta(q, c, A) = \emptyset \)
(11) \( \delta(q, c, B) = \emptyset \)
(12) \( \delta(q, c, C) = \{(q, AB), (q, C)\} \) \( C \rightarrow cAB|cC \)
(13) \( \delta(q, \varepsilon, S) = \emptyset \)
(14) \( \delta(q, \varepsilon, A) = \emptyset \)
(15) \( \delta(q, \varepsilon, B) = \emptyset \)
(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.
  – What if ε is in L?

• **Suppose ε is in L:**

  1) First, let $L' = L - \{\varepsilon\}$

     By an earlier theorem, if L is a CFL, then $L' = L - \{\varepsilon\}$ 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 $\delta(q, \varepsilon, S) = \{(q, \varepsilon)\}$? No!
• Counter Example:

Consider \( L = \{ \epsilon, b, ab, aab, aaab, \ldots \} \)

Then \( L' = \{ b, ab, aab, aaab, \ldots \} \)

• The GNF CFG for \( L' \):

(1) \( S \rightarrow aS \)
(2) \( S \rightarrow b \)

• The PDA \( M \) Accepting \( L' \):

\[
\begin{align*}
Q &= \{ q \} \\
\Sigma = T &= \{ a, b \} \\
\Gamma = V &= \{ S \} \\
\zeta &= S \\
\delta(q, a, S) &= \{ (q, S) \} \\
\delta(q, b, S) &= \{ (q, \epsilon) \} \\
\delta(q, \epsilon, S) &= \emptyset
\end{align*}
\]

• If \( \delta(q, \epsilon, S) = \{ (q, \epsilon) \} \) is added then:

\( L(M) = \{ \epsilon, a, aa, aaa, \ldots, b, ab, aab, aaab, \ldots \} \)
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 $L = L(G)$ if and only if there exists a PDA $M$ such that $L = L_E(M)$.

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