Chapter 7

PDA and CFLs
7.1 PDA

- Is an enhanced FSA with an **internal memory** system, i.e., a *(pushdown) stack.*

- Overcomes the memory limitations and **increases** the *processing power* of FSAs.

**Defn. 7.1.1** A pushdown automaton (PDA) is a sextuple \((Q, \Sigma, \Gamma, \delta, q_0, F)\), where

- \(Q\) is a finite set of **states**
- \(\Sigma\) is a finite set of input symbols, called **input alphabet**
- \(\Gamma\) is a finite set of stack symbols, called **stack alphabet**
- \(q_0 \in Q\), is the **start state**
- \(F \subseteq Q\), is the set of **final states**
- \(\delta: Q \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \rightarrow Q \times (\Gamma \cup \{\lambda\})\), a *(partial) transition function*
7.1 PDA

A convention:

- Stack symbols are *capital letters*
- Greek letters represent *strings of stack symbols*
- An empty stack is denoted \( \lambda \)
- \( A\alpha \) represents a stack with \( A \) as the *top element*
7.1 PDA

- $\delta$ is of the form

$$\delta(q_{i_0}, a, A_0) = \{ [q_{i_1}, A_1], [q_{i_2}, A_2], \ldots, [q_{i_n}, A_n] \}$$

where the transition

$$[q_{i_j}, A_j] \in \delta(q_{i_0}, a, A_0), \ 1 \leq j \leq n$$

denotes that

- $q_{i_0}$ is the current state
- $a$ is the current input symbol
- $A_0$ is the current top of the stack symbol
- $q_{i_j} \ (1 \leq j \leq n)$ is the new state, and
- $A_j$ is the new top of the stack symbol and in a state (transition) diagram, it is denoted

$q_{i_0} \ a \ A_0/A_j \ q_{i_j}$
7.1 Pushdown Automaton

- Input Tape
- Tape Head
- Head moves in this direction
- Control Mechanism
- Stack

Diagram showing a state indicator with states $s_0, s_1, s_2, s_3, s_4, s_5$. The tape head moves in the indicated direction.
7.1 PDA

Special cases (note that $q_i$ and $q_j$ can be the same):

- $[q_j, A] \in \delta(q_i, a, \lambda)$ /* Consume the input, push a stack symbol */
- $[q_j, \lambda] \in \delta(q_i, \lambda, A)$ /* Consume no input, pop the TOS symbol */
- $[q_j, A] \in \delta(q_i, \lambda, \lambda)$ /* Consume no input, push a stack symbol */
- $[q_j, \lambda] \in \delta(q_i, a, \lambda)$ /* Consume input, no push/pop, an FSA transition */

The PDA notation $[q_i, w, \alpha] \xrightarrow{*}{m} [q_j, v, \beta]$ indicates that $[q_j, v, \beta]$ can be obtained from $[q_i, w, \alpha]$ as a result of a sequence of transitions (i.e., 0 or more).

Example. The PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, where $\delta$ is

Accepts $a^n b^n (n \geq 0)$ with an empty stack and in an accepting state
7.1 PDA

- **Defn. 7.1.2** Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a PDA. A string $w \in \Sigma^*$ is accepted by $M$ if $\exists [q_0, w, \lambda] \xrightarrow{*} [q_i, \lambda, \lambda]$, where $q_i \in F$.

$L(M)$, the language of $M$, is the set of strings accepted by $M$.

- **Example (7.1.1)** Give a PDA that accepts the language $\{ wcw^R \mid w \in \{ a, b \}^* \}$.

- **Defn.** A PDA is *deterministic* if there is at most one transition that is applicable for each configuration of $<state, input symbol, stack top symbol>$.

  - **Example 7.1.2** Construct $L = \{ a^i \mid i \geq 0 \} \cup \{ a^i b^i \mid i \geq 0 \}$

  - **Example 7.1.3** Construct all even-length palindromes over $\{ a, b \}$

- Nondeterministic PDAs allow the machines to “guess”
- For some NPDAs, their counterparts (i.e., DPDAs) do not exist
- Languages $L$ accepted by DPDA include $RL$, and $L \subset CFL$
7.2 Variations on PDAs

- **Defn.** A PDA is *atomic* if each transition in the PDA is of one of the following forms:
  
  \[
  \begin{align*}
  [q_j, \lambda] &\in \delta(q_i, a, \lambda) &\text{: process an input symbol} \\
  [q_j, \lambda] &\in \delta(q_i, \lambda, A) &\text{: pop the stack} \\
  [q_j, A] &\in \delta(q_i, \lambda, \lambda) &\text{: push a stack symbol}
  \end{align*}
  \]

- **Theorem 7.2.1.** Let $M$ be a PDA. Then $\exists$ an atomic PDA $M'$ such that $L(M') = L(M)$.
  
  - Replace each non-atomic transition by a sequence of atomic transitions.

- **Defn.** A transition $[q_j, \alpha] \in \delta(q_i, a, A)$, where $\alpha \in \Gamma^+$ is called *extended transition*. A PDA containing extended transition is called an *extended PDA*.
  
  - **Example 7.2.1.** Construct $L = \{ a^i b^{2i} \mid i \geq 1 \}$, with PDA, atomic PDA, and extended PDA.
Example 7.2.1

Let $L = \{a^i b^{2i} \mid i \geq 1\}$. A PDA, an atomic PDA, and an extended PDA are constructed to accept $L$. The input alphabet $\{a, b\}$, stack alphabet $\{A\}$, and accepting state $q_1$ are the same for each automaton.

**PDA**

- $Q = \{q_0, q_1, q_2\}$
- $\delta(q_0, a, \lambda) = \{[q_2, A]\}$
- $\delta(q_2, \lambda, \lambda) = \{[q_0, A]\}$
- $\delta(q_0, b, A) = \{[q_1, \lambda]\}$
- $\delta(q_1, b, A) = \{[q_1, \lambda]\}$

**Atomic PDA**

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\delta(q_0, a, \lambda) = \{[q_3, \lambda]\}$
- $\delta(q_3, \lambda, \lambda) = \{[q_2, A]\}$
- $\delta(q_2, \lambda, \lambda) = \{[q_0, A]\}$
- $\delta(q_0, b, \lambda) = \{[q_4, \lambda]\}$
- $\delta(q_4, \lambda, A) = \{[q_1, \lambda]\}$
- $\delta(q_1, b, \lambda) = \{[q_4, \lambda]\}$

**Extended PDA**

- $Q = \{q_0, q_1\}$
- $\delta(q_0, a, \lambda) = \{[q_0, AA]\}$
- $\delta(q_0, b, A) = \{[q_1, \lambda]\}$
- $\delta(q_1, b, A) = \{[q_1, \lambda]\}$
Defn. A string $w$ is accepted by final state if $\exists$ a computation $[q_0, w, \lambda] \xrightarrow{*} [q_i, \lambda, \alpha]$, where $q_i \in F$ and $\alpha \in \Gamma^*$, i.e., the content of the stack is irrelevant.

Lemma 7.2.3. Let $L$ be a language accepted by a PDA $M$ with acceptance defined by final state. Then $\exists$ a PDA $M'$ that accepts $L$ by final state and empty stack.

Proof. Let $M' = (Q \cup \{q_f\}, \Sigma, \Gamma, \delta', q_0, \{q_f\})$, where $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, $\delta' \supseteq \delta$, $\forall q_i \in F, \delta'(q_i, \lambda, \lambda) = \{ [q_f, \lambda] \}$, and $\forall A \in \Gamma, \delta'(q_f, \lambda, A) = \{ [q_f, \lambda] \}$.

hence, given $M$,

$[q_0, w, \lambda] \xrightarrow{M} [q_i, \lambda, \alpha] \xrightarrow{M} [q_f, \lambda, \alpha] \xrightarrow{M} [q_f, \lambda, \lambda]$. 
PDA

- **Defn.** A string \( w \) is said to be accepted by empty stack if \( \exists \) a computation \([q_0, w, \lambda] \xrightarrow{\pm} [q_i, \lambda, \lambda]\), where \( q_i \) may not be a final state.

- **Lemma 7.2.4.** Let \( L \) be a language accepted by a PDA \( M \) with acceptance defined by empty stack. Then \( \exists \) a PDA \( M' \) that accepts \( L \) by final state and empty stack.

**Proof.** (P. 230) Let \( M' = (Q \cup \{q_0'\}, \Sigma, \Gamma, \delta', q_0', Q) \), where \( M = (Q, \Sigma, \Gamma, \delta, q_0) \), and each state in \( M \) is a final state, except \( q_0' \), the new start state, where

\[
\delta'(q_0', a, A) = \delta(q_0, a, A), \text{ and } \\
\forall q_i \in Q, a \in \Sigma \cup \{\lambda\}, A \in \Gamma \cup \{\lambda\}, \quad \delta'(q_i, a, A) = \delta(q_i, a, A)
\]
Two-Stack PDAs

- Two-stack PDAs, an extension of PDAs
- A two-stack PDA (2PDA) is a sextuple \((Q, \sum, \Gamma, \delta, q_0, F)\), where
  - \(Q, \sum, \Gamma, q_0,\) and \(F\) are the same as in a one-stack PDA
  - \(\delta: Q \times (\sum \cup \{ \lambda \}) \times (\Gamma \cup \{ \lambda \}) \times (\Gamma \cup \{ \lambda \}) \rightarrow Q \times (\Gamma \cup \{ \lambda \}) \times (\Gamma \cup \{ \lambda \})\)
- 2PDAs accept non-CFLs, in addition to all CFLs
- Accepting criteria:
  - Consume an *input string*
  - Enter a *final state*
  - *Empty both stacks*
Two-Stack PDAs

Example. Given $L = \{ a^i b^i c^i \mid i \geq 0 \}$, $L$ is not a CFL.

A 2PDA $M$ that accepts $L$ is
Two-Stack PDAs

Example. A 2PDA $M$ accepts $L = \{ a^i b^i c^i d^i \mid i \geq 0 \}$

- $a \lambda/A \lambda/\lambda \xrightarrow{} q_0$
- $\lambda \lambda/\lambda/\lambda \xrightarrow{} q_3$
- $b A/\lambda/\lambda/B \xrightarrow{} q_1$
- $\lambda \lambda/\lambda/\lambda \xrightarrow{} q_2$
- $c \lambda/C B/\lambda \xrightarrow{} q_1$
- $d C/\lambda/\lambda/\lambda \xrightarrow{} q_3$

Transitions:

- $[q_0, aabbccdd, \lambda, \lambda] \xrightarrow{} [q_0, abbcddd, A, \lambda]$
- $[q_0, bbccdd, AA, \lambda]$
- $[q_1, bccdd, A, B]$
- $[q_1, ccdd, \lambda, BB]$
- $[q_2, cdd, C, B]$
- $[q_2, dd, CC, \lambda]$
- $[q_3, d, C, \lambda]$
- $[q_3, \lambda, \lambda, \lambda]$
7.3 PDA and CFLs

- **Defn. 5.6.1.** A CFG $G = (V, \Sigma, P, S)$ is in **Greibach normal form (GNF)** if each rule has one of the following forms:
  
  i) $A \rightarrow aA_1A_2 \ldots A_n$
  
  ii) $A \rightarrow a$
  
  iii) $S \rightarrow \lambda$

  where $a \in \Sigma$ and $A_i \in V - \{S\}$, $i = 1, 2, \ldots, n$

- **Example:** Given the language $L = \{ a^i b^i c^k \mid i, j, k \geq 0 \text{ and } (i = j \text{ or } i = k) \}$, the following PDA accepts $L$:
The CFG $G$ that generates the set of string in the language

$L = \{ ai^jb^jck | i, j, k \geq 0 \text{ and } (i = j \text{ or } i = k) \}$, i.e., $L(G)$, is

$$
S \rightarrow aAc \mid aDbC \mid \lambda \mid B \mid C
$$

$i = k$ \{ 
  
  $$A \rightarrow aAc \mid bB \mid \lambda
  $$
  
  $$B \rightarrow bB \mid \lambda
  $$
\}

$i = j$ \{ 
  
  $$D \rightarrow aDb \mid \lambda
  $$
  
  $$C \rightarrow cC \mid \lambda
  $$
\}
7.3 PDA and CFLs

Theorem 7.3.1 Let $L$ be a CFL. Then $\exists$ a PDA that accepts $L$.

Proof. Let $G = (V, \Sigma, P, S)$ be a grammar in GNF that generates $L$. An extended PDA $M$ with start state $q_0$ is defined by

$$Q_m = \{ q_0, q_1 \}, \Sigma_m = \Sigma, \Gamma_m = V - \{ S \}, \text{ and } F_m = \{ q_1 \}$$

with transitions

(a) $\delta(q_0, a, \lambda) = \{ [q_1 , w] \mid S \rightarrow aw \in P \}$

(b) $\delta(q_1, a, A) = \{ [q_1 , w] \mid A \rightarrow aw \in P \text{ and } A \in V - \{ S \} \}$

(c) $\delta(q_0, \lambda, \lambda) = \{ [q_1 , \lambda] \mid S \rightarrow \lambda \in P \}$
**Proof.** We must show that

i) \( L \subseteq L(M) \)

For each derivation \( S \Rightarrow uw \) with \( u \in \Sigma^+ \) and \( w \in V^* \), we show that \( \exists \) a computation \([q_0, u, \lambda] \xrightarrow{\ast} [q_1, \lambda, w]\) in \( M \) by induction on the length of the derivation, i.e., \( n \).

**Basis:** \( n = 1 \), i.e., \( S \Rightarrow aw \), where \( a \in \Sigma \) and \( w \in V^* \)

The transition \((a)\), i.e., \( \delta(q_0, a, \lambda) = \{ [q_1, w] | S \rightarrow aw \in P \}\), yields the desired computation.

**Induction Hypothesis:**

Assume for every derivation \( S \Rightarrow uvw \), \( \exists \) a computation \([q_0, u, \lambda] \xrightarrow{\ast} [q_1, \lambda, w]\) in \( M \).
Induction:

Now consider \( S^{n+1} uw \). Let \( u = va \in \Sigma^+ \) & \( w \in V^* \), \( S^{n+1} uw \) can be written as \( S \Rightarrow vAw_2 \Rightarrow uw \), where \( w = w_1w_2 \) & \( A \Rightarrow aw_1 \in P \).

By I.H. & \([q_1, w_1] \in \delta(q_1, a, A) \) of Transition (b), i.e.,

\[
\delta(q_1, a, A) = \{ [q_1, w] | A \rightarrow aw \in P \text{ & } A \in V - \{ S \} \}
\]

\[
[q_0, va, \lambda] \xrightarrow{*} [q_1, a, Aw_2] \\
\xrightarrow{} [q_1, \lambda, w_1w_2]
\]

If \( \lambda \in L \), then \( S \rightarrow \lambda \in P \) yields the Transition (c), i.e.,

\[
[q_0, \lambda, \lambda] \longrightarrow [q_1, \lambda]
\]

ii) \( L(M) \subseteq L \)

Show that for every computation \([q_0, u, \lambda] \xrightarrow{*} [q_1, \lambda, w]\), \( \exists \) a derivation \( S \Rightarrow uw \) in \( G \) by induction.
PDA and CFLs

- Every language accepted by a PDA is context-free

- The *Transformation Algorithm*: Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a PDA. We construct the grammar $G \vdash L(G) = L(M)$.

  (i) Construct an extended PDA $M'$ w/ $\delta'$ as its transition function from $M \vdash$.

  (a) Given $[q_j, \lambda] \in \delta(q_i, u, \lambda)$, construct $[q_j, A] \in \delta'(q_i, u, A)$, $\forall A \in \Gamma$

  (b) Given $[q_j, B] \in \delta(q_i, u, \lambda)$, construct $[q_j, BA] \in \delta'(q_i, u, A)$, $\forall A \in \Gamma$

  (ii) Given the PDA $M'$ as constructed in step (i), construct $G = (V, \Sigma, P, S)$, where $V = \{ S \} \cup \{ <q_i, A, q_j> \mid q_i, q_j \in Q$, $A \in \Gamma \cup \{ \lambda \} \}$. $<q_i, A, q_j>$ denotes a computation that begins in $q_i$, ends in $q_j \&$ removes $A \in \Gamma$ from the stack.
The Transformation Algorithm

- $P$ is constructed as follows:

1. $S \rightarrow <q_0, \lambda, q_j>$, $\forall q_j \in F$

2. For each transition $[q_j, B] \in \delta(q_i, X, A)$, where $A \in \Gamma \cup \{\lambda\}$, create $\{<q_i, A, q_k> \rightarrow X<q_j, B, q_k> | q_k \in Q\}$

3. For each transition $[q_j, BA] \in \delta(q_i, X, A)$, where $A \in \Gamma$, create $\{<q_i, A, q_k> \rightarrow X<q_j, B, q_n><q_n, A, q_k>| q_k, q_n \in Q\}$

4. For each $q_k \in Q$, create $<q_k, \lambda, q_k> \rightarrow \lambda$.

Rule 1: A computation begins w/ the start state, ends in a final state, & terminate w/ an empty stack, i.e., a successful computation in $M'$

Rules 2 & 3: Trace the transitions of $M'$

Rule 4: Terminate derivations
Example 7.3.1. Given the PDA $M$ such that $L(M) = \{ a^n c b^n | n \geq 0 \}$. The corresponding CFG $G$ is given in Table 7.3.1 (on P.240).

$M$ is

$Q = \{ q_0, q_1 \}$  \hspace{1cm} $\delta(q_0, a, \lambda) = \{ [q_0, A] \}$

$\Sigma = \{ a, b, c \}$  \hspace{1cm} $\delta(q_0, c, \lambda) = \{ [q_1, \lambda] \}$

$\Gamma = \{ A \}$  \hspace{1cm} $\delta(q_1, b, A) = \{ [q_1, \lambda] \}$

$F = \{ q_1 \}$

$M'$ is $M$ with the additional transitions:

$\delta(q_0, a, A) = \{ [q_0, AA] \}$ and

$\delta(q_0, c, A) = \{ [q_1, A] \}$
Example 7.3.1 $P$ in $G$ includes:

- using Rule 1: $S \rightarrow < q_0, \lambda, q_1 >$
- given $\delta(q_0, a, \lambda) = \{ [q_0, A] \}$ & Rule 2:
  $< q_0, \lambda, q_0 > \rightarrow a < q_0, A, q_0 >$
  $< q_0, \lambda, q_1 > \rightarrow a < q_0, A, q_1 >$

- given $\delta(q_0, a, A) = \{ [q_0, AA] \}$ & Rule 3:
  $< q_0, A, q_0 > \rightarrow a < q_0, A, q_0 > < q_0, A, q_0 >$
  $< q_0, A, q_0 > \rightarrow a < q_0, A, q_1 > < q_1, A, q_0 >$
  $< q_0, A, q_1 > \rightarrow a < q_0, A, q_1 > < q_1, A, q_1 >$

- given $\delta(q_0, c, \lambda) = \{ [q_1, \lambda] \}$ & Rule 2:
  $< q_0, \lambda, q_0 > \rightarrow c < q_1, \lambda, q_0 >$
  $< q_0, \lambda, q_1 > \rightarrow c < q_1, \lambda, q_1 >$

- given $\delta(q_0, c, A) = \{ [q_1, A] \}$ & Rule 2:
  $< q_0, A, q_0 > \rightarrow c < q_1, A, q_0 >$
  $< q_0, A, q_1 > \rightarrow c < q_1, A, q_1 >$

- given $\delta(q_1, b, A) = \{ [q_1, \lambda] \}$
  $< q_1, A, q_0 > \rightarrow b < q_1, \lambda, q_0 >$
  $< q_1, A, q_1 > \rightarrow b < q_1, \lambda, q_1 >$

- using Rule 4:
  $< q_0, \lambda, q_0 > \rightarrow \lambda$
  $< q_1, \lambda, q_1 > \rightarrow \lambda$