This lecture covers Chapter 6 of HMU: Pushdown Automata

- Pushdown Automata (PDA)
- Language accepted by a PDA
- Equivalence of CFGs and the languages accepted by PDAs
- Deterministic PDAs

Additional Reading: Chapter 6 of HMU.
Introduction to PDAs

- PDA ‘=’ ε-NFA + Stack (LIFO)
- At each instant, the PDA can choose to read a symbol or not.
- Transitions depend on a subset of: (a) the input symbol, if read; (b) present state; and (c) symbol atop the stack.
- At each instant, a transition can potentially induce some or all of the following: (a) change in state; (b) push a string or pop a symbol from the stack.
- Once the string is read, the PDA decides to accept/reject the input string.
- Note: The PDA can only read a symbol once (i.e., the reading head is unidirectional).
A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where

- $Q$ is the (finite) set of internal states; $\Sigma$ is the finite alphabet of input tape symbols; $q_0 \in Q$ is the (unique) start state; $F$ is the set of final or accepting states of the PDA.
- $\Gamma$ is the finite alphabet of stack symbols;
- $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$ (power set of $Q \times \Gamma^*$) such that $\delta(q, a, \gamma)$ is always a finite set of pairs $(q', \gamma') \in Q \times \Gamma^*$.
- $Z_0 \in \Gamma$ is the sole symbol atop the stack at the start; and

Convention: lower case symbols $s$, $a$, and $b$ will denote input symbols; lower case symbols $u$, $v$, $w$ will exclusively denote strings of input symbols; stack symbols are indicated by upper case letters (e.g., $A$, $B$, etc); strings of stack symbols are indicated by greek letters (e.g., $\alpha$, $\beta$, etc);
A PDA Example

Transition Diagram Notation

Notation: The label $a, A/\gamma$ on the edge from a state $q$ to $q'$ indicates a possible transition from state $q$ to state $q'$ by reading the symbol $a$ when the top of the stack contains the symbol $A$. This stack symbol is then replaced by the string $\gamma$.

$$(q', \gamma) \in \delta(q, a, A) \iff \begin{array}{c}
\begin{array}{c}
q \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
a, A/\gamma \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
q'
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
(Note: q' can be q itself)
\end{array}
\end{array}$$

PDA that accepts $L = \{ww^R : w \in \{0,1\}^*\}$

$\begin{array}{c}
\begin{array}{c}
q_0 \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
\epsilon, Z_0/Z_0 \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
q_1 \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
\epsilon, 0/0 \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
\epsilon, 1/1 \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
\epsilon, Z_0/Z_0 \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
\epsilon, 1/0 \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
\epsilon, 1/1 \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
0, 0/\epsilon \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
1, 1/\epsilon \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
0, Z_0/0Z_0 \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
1, Z_0/1Z_0 \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
0, 0/00 \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
1, 0/10 \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
0, 1/01 \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
1, 1/11 \\
\rightarrow
\end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
q_2
\end{array}
\end{array}$$
Language Accepted by a PDA

Definitions

> The **Configuration** or **Instantaneous Description (ID)** of a PDA $P$ is a triple $(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$ where:

(i) $q$ is the state of the PDA;
(ii) $w$ is the unread part of input string; and
(iii) $\gamma$ is the stack contents from top to bottom.

> An ID tracks the trajectory/operation of the PDA as it reads the input string.

> One-step computation of a PDA is based on a single move/transition (change of IDs). Suppose $(q', \gamma) \in \delta(q, a, A)$. Then for any $w \in \Sigma^*$, $\alpha \in \Gamma^*$,

$$(q, aw, A\alpha) \vdash_P (q', w, \gamma\alpha), \quad \text{[one-step computation]}$$

> We denote (multi-step) computation by $\vdash_P^*$, which indicates zero, or any finite number of consecutive PDA transitions. We denote $\text{ID} \vdash_P^* \text{ID}'$ if there are $k$ IDs $\text{ID}_1, \ldots, \text{ID}_k$ (for some $k \geq 2$) such that:

(i) $\text{ID}_1 = \text{ID}$ and $\text{ID}_k = \text{ID}'$, and
(ii) for each $i = 1, \ldots, k - 1$, either $\text{ID}_i = \text{ID}_{i+1}$ or $\text{ID}_i \vdash_P \text{ID}_{i+1}$.
Beware of PDAs and IDs!

Lemma 6.2.1

Let PDA \( P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \) be given. Let \( q, q' \in Q \), \( x, y, w \in \Sigma^* \), and \( \alpha, \beta, \gamma \in \Sigma^* \). Then the following hold.

\[
(q, x, \alpha) \xrightarrow[\text{P}]{} (q, y, \beta) \iff (q, xw, \alpha) \xrightarrow[\text{P}]{} (q, yw, \beta) \quad (1)
\]

\[
(q, x, \alpha) \xrightarrow[\text{P}]{} (q, y, \beta) \implies (q, x, \alpha\gamma) \xrightarrow[\text{P}]{} (q, y, \beta\gamma) \quad (2)
\]

Proof Idea

> The equivalence in (1) simply follows from the fact that reading the input is unidirectional. If you have just finished reading \( x \) or are yet to, then what follows \( x \) can simply not affect your past transitions/configurations.

> PDA transitions occur only when the stack is non-empty. If \((q, x, \alpha) \xrightarrow[\text{P}]{} (q, y, \beta)\), then the transitions that effect that ID change could have never emptied the stack (at any intermediate step). A simple proof based on induction (on the number of transitions/ID changes) along with the fact that the stack is never emptied completes the claim.
Language Accepted by PDAs

**Definition**

Given PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, the language accepted by $P$ by final states is

$$L(P) = \left\{ w \in \Sigma^* : (q_0, w, Z_0) \xrightarrow{\star}_P (q, \epsilon, \alpha) \text{ for some } q \in F, \alpha \in \Gamma^* \right\}.$$  

The language accepted by $P$ by empty stack is

$$N(P) = \left\{ w \in \Sigma^* : (q_0, w, Z_0) \xrightarrow{\star}_P (q, \epsilon, \epsilon) \text{ for some } q \in Q \right\}.$$  

**Can $L(P)$ and $N(P)$ be different?**

> Pick a DFA $A$ such that $L(A) \neq \emptyset$. Convert it to a PDA $P$ by pushing each symbol that is read onto the stack, increasing the stack size each time a symbol is read. The PDA has never pops a stack symbol. For the derived PDA, $L(P) = L(A)$. However, $N(P) = \emptyset$.

> Which of the two definitions accepts 'more' languages?
Equivalence of the Two Notions of Language Acceptance

**Theorem 6.2.2**

Given PDA $P$, there exist PDAs $P'$ and $P''$ such that $L(P) = N(P')$ and $N(P) = L(P'')$.

**Proof of Existence of $P''$**

- Introduce a new start state and a new final state with the transitions as indicated.
- The start state first replaces the stack symbol $Z_0$ by $Z_0X_0$.
- If and only if $w \in N(P)$ will the computation by $P$ end with the stack containing precisely $X_0$.
- The PDA $P''$ then transitions to the final state popping $X_0$. Hence, $N(P) = L(P'')$. 
Equivalence of the two Notions of Language Acceptance

Proof of Existence of $P'$

> Introduce a new start state and a special state with the transitions as indicated.

> The start state first replaces the stack symbol $Z_0$ by $Z_0X_0$.

> If and only if $w \in L(P)$ will the computation by $P$ end in a final state with the stack containing (at least) $X_0$.

> The PDA $P'$ then transitions to the special state and starts to pop stack symbols one at a time until the stack is empty. Hence, $L(P) = N(P')$. 
Theorem 6.3.1

For every CFG $G$, there exists a PDA $P$ such that $N(P) = L(G)$.

Proof

$\triangleright$ Let $G = (V, T, P, S)$ be given.

$\triangleright$ Construct PDA $P = (\{q_0\}, V, V \cup T, \delta, S, \{q_0\})$ with $\delta$ defined by

[Type 1] $\delta(q_0, a, a) = \{(q_0, \epsilon)\}$, whenever $a \in \Sigma$,

[Type 2] $\delta(q_0, \epsilon, A) = \{(q_0, \alpha) : A \longrightarrow \alpha$ is a production rule in $P\}$.

$\triangleright$ This PDA mimics all possible leftmost derivations.

$\triangleright$ We use induction to show that $L(G) = N(P)$.
Proof of 1-1 Correspondence between PDA Moves and Leftmost Derivations

Suppose $w \in T^*$ and $S \xrightarrow{LM}^* w$.

Let $w_i \in T^*$, $V_i \in V$, $\alpha_i \in (V \cup T)^*$.

Leftmost Derivation in Grammar $G$

- $S \xrightarrow{\gamma_1} w_2 V_2 \alpha_2$
- $V_2 \xrightarrow{\gamma_2} w_3 V_3 \alpha_3$
- $V_3 \xrightarrow{\gamma_3} w_4 V_4 \alpha_4$
- $V_4 \xrightarrow{\gamma_4} \ldots$
- $w_k = w$

Unread Part of Input Tape

- Stack
- Stack Symbols that have been popped

$A \setminus B :=$ The suffix of $B$ in $A$
Theorem 6.3.2
For every PDA $P$, there exists a CFG $G$ such that $L(G) = N(P)$.

Proof

- Given $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$, we define $G = (V, T, P, S)$ that mimics the computations of $P$ as follows.
  - $T = \Sigma$;
  - $V = \{S\} \cup \{[pXq] : p, q \in Q, X \in \Gamma\}$;
    Interpretation: Each variable $[pXq]$ will generate a terminal string $w$ in $G$ iff if $w$ induces a move (in finite steps) from the state $p$ to $q$ popping $X$ from the stack.
  - $P$ contains only the following rules:
    - $S \rightarrow [q_0Z_0p]$ for all $p \in Q$.
    - Suppose that $(r, X_1 \cdots X_\ell) \in \delta(q, a, X)$. Then, for any states $p_1, \ldots, p_\ell \in Q$,
      $$[qXp_k] \rightarrow a[rX_1p_1][p_2X_2p_2] \cdots [p_{\ell-1}X_\ell p_\ell].$$
      Note that if $(r, \epsilon) \in \delta(q, a, X)$, then $[qXp_k] \rightarrow a$.  


Proof of \((q, w, X) \vdash_P^* (p, \epsilon, \epsilon) \Rightarrow [qXp] \vdash_G^* w\). (Induction on \# of steps of computation)

- **Basis:** Let \(w \in N(P)\). Suppose there is a one-step computation \((q, w, X) \vdash_P (p, \epsilon, \epsilon)\). Then, \(w \in \Sigma \cup \{\epsilon\}\). Since \((p, \epsilon) \in \delta(q_0, w, X)\), \([q_0Xp] \rightarrow w\) is a production rule.

- **Induction:** Let \((q, w, X) \vdash_P^* (p, \epsilon, \epsilon)\). Let \(a\) be read in the first step of the computation, and let \(w = ax\). Then the following argument completes the proof.
**CFGs and PDAs**

Proof of $[qXp] \xrightarrow{\ast} w \Rightarrow (q, w, X) \xrightarrow{\ast} (p, \epsilon, \epsilon)$. (Induction on \# of steps of derivation)

**Basis:** Let $[qXp] \xrightarrow{\ast} w$ in one step. Then, $[qXp] \rightarrow w$ must be a production rule. Consequently, $(p, \epsilon) \in (q, w, X)$ and $(q, w, X) \xrightarrow{P} (p, \epsilon, \epsilon)$.

**Induction:** Let $[qXp] \xrightarrow{P} w$.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$[qXp] \xrightarrow{LM} a[r_0Y_1] [r_1Y_2] \cdots [r_{k-1}Y_k p]$ $\xrightarrow{LM} w = aw_1 \cdots w_k$</td>
</tr>
<tr>
<td>2</td>
<td>Induc. for $w_1$ and $w_2$</td>
</tr>
<tr>
<td>3</td>
<td>Induc. for $(r_0, w_1, Y_1) \xrightarrow{P} (r_1, \epsilon, \epsilon)$ and $(r_1, w_2, Y_2) \xrightarrow{P} (r_2, \epsilon, \epsilon)$</td>
</tr>
<tr>
<td>4</td>
<td>$(r_0, Y_1 \cdots Y_k) \in \delta(q, a, X)$ $\iff (q, a, X) \xrightarrow{P} (r_0, \epsilon, Y_1 \cdots Y_k)$</td>
</tr>
<tr>
<td>5</td>
<td>Lemma 6.2.1</td>
</tr>
</tbody>
</table>
Deterministic PDAs (DPDAs)

> PDAs are (by definition) non-deterministic.
> Deterministic PDAs are defined to have no choice in their transitions.

**Definition**

A DPDA $P$ is a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ such that for each $q \in Q$ and $X \in \Gamma$,

> $|\delta(q, a, X)| \leq 1$ for any $a \in \Sigma \cup \{\epsilon\}$,
  i.e., a configuration cannot transition to more than one configuration.

> $|\delta(q, a, X)| = 1$ for some $a \in \Sigma \Rightarrow \delta(q, \epsilon, X) = \emptyset$,
  i.e., both reading or not reading (a tape symbol) cannot lead to valid configurations.

> DPDAs have a computation power that is strictly better than DFAs

**Example:** $L(P) = N(P) = \{0^n1^n : n \geq 1\}$

> DPDAs have a computation power that is strictly worse than PDAs.
  (We will discuss this later)
Languages Accepted by DPDAs

- The two notions of acceptance (empty stack and final state) are not equivalent in the case of DPDAs.
- There are languages \( L \) such that \( L = L(P) \) for some DPDA \( P \), but there exists no \( P' \) such that \( L = N(P') \).

**Theorem 6.4.1**

Every regular language \( L \) is the language accepted by the final states of some DPDA.

**Proof**

Simply view the DFA accepting \( L \) as a DPDA (with the stack always containing \( Z_0 \)).

- The regular language \( L = \{0\}^* \) cannot equal \( N(P) \) for any DPDA \( P \).
  - Suppose DPDA \( P \) accepts \( L \) by emptying its stack. Since 0 is accepted, \( P \) eventually reaches a configuration \((p, \epsilon, \epsilon)\) for some state \( p \).

  Now, suppose that \( P \) is fed with the input 00. Since \( P \) is deterministic, \( P \) reads a 0 and eventually has to get to \((p, \epsilon, \epsilon)\). However, it hangs at this configuration and cannot read any further input symbols. Hence, \( P \) cannot accept 00.
Languages Accepted by DPDAs

> A language $L$ is said to have the **prefix property** if no two distinct strings in the language are prefixes of one another.

**Theorem 6.4.2**

A language $L = N(P)$ for some DPDA $P$ iff $L$ has the prefix property and $L = L(P'')$ for some DPDA $P''$.

**Proof**

⇒ Let $L = N(P)$ for some DPDA $P$. Let $w, ww'$ be in $L$ with $w' \neq \epsilon$. Then $(q_0, w, Z_0) \xrightarrow{\ast} (p, \epsilon, \epsilon)$ for some $p \in Q$. The DPDA hangs at this state since the stack is empty. Hence, it cannot accept $ww'$. The fact that $L = L(P'')$ for some DPDA $P''$ follows from Theorem 6.2.2 since the construction yields a **deterministic** PDA.
Languages Accepted by DPDAs

Proof

\[ \iff \text{Let DPDA } P'' \text{ be given. Let } w \in L(P''), (q_0, w, Z_0) \xrightarrow{*}_P (p, \epsilon, \gamma) \text{ for some } p \in F, \text{ and } \gamma \in \Gamma. \text{ Since } L(P'') \text{ satisfies the prefix property, it must be true that the configurations in-between } (q_0, w, Z_0) \text{ and } (p, \epsilon, \gamma) \text{ only pass through non-final states.} \]

\[ \implies \text{Thus, redefining } \delta(p, a, X) = \emptyset \text{ for all } p \in Q, a \in \Sigma \text{ and } X \in \Gamma \text{ does not alter } L(P''). \]

\[ \implies \text{Then, the construction of Theorem 6.2.2 yields a deterministic PDA } P' \text{ such that } N(P') = L(P'') = L. \]
DPDAs and Unambiguous Grammars

Theorem 6.4.3

If \( L = N(P) \) for some DPDA \( P \), then \( L \) has an unambiguous CFG.

Proof

> Let \( G \) be the CFG constructed in Theorem 6.3.2.

> Suppose \( G \) is ambiguous. Then, for some \( w \in L \) has 2 leftmost derivations.

> However, each derivation corresponds to a unique trajectory of configurations in \( P \) that also accepts \( w \) by emptying stack.

> Since \( P \) is deterministic, the trajectories, and hence, the derivations have to be identical. Hence, \( G \) is unambiguous.
Deterministic PDAs

DPDAs and unambiguous Grammars

Theorem 6.4.4

If \( L = L(P) \) for some DPDA \( P \), then \( L \) has an unambiguous CFG.

Proof

\( \triangleright \) Let $ be a symbol not in the alphabet of \( L \).

\( \triangleright \) Consider \( L' = \{ w$ : \( w \in L \} \). Then, \( L' \) has the prefix property.

\( \triangleright \) By Theorem 6.4.2, there must exist a DPDA \( P' \) such that \( L' = N(P') \).

\( \triangleright \) By Theorem 6.4.3, \( L' \) has an unambiguous CFG \( G' = (V, T, P, S) \).

\( \triangleright \) Define CFG \( G = (V \cup \{\}$, \( T \setminus \{\}$, \( P \cup \{\$ \rightarrow \epsilon\}$, \( S) \).

\( \triangleright \) \( G \) generates \( L \).

\( \triangleright \) Suppose \( G \) is ambiguous. Then, for some \( w \in L \) has 2 leftmost derivations.

\( \triangleright \) The last steps in the two leftmost derivations of \( w \) must use the production \( \$ \rightarrow \epsilon \).

\( \triangleright \) Then, the portions of the two leftmost derivations without the last production step correspond to two leftmost derivations of \( w\$ \).

\( \triangleright \) Hence, \( G' \) must be unambiguous, which is a contradiction. Hence, \( G \) is also unambiguous.