Recap: Definition of PDA

A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where

- $Q$ is a finite set of states
- $\Sigma$ is a finite set of input symbols
- $\Gamma$ is a finite set of stack symbols
- $\delta \in Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow P_{\text{id}}(Q \times \Gamma^*)$ is the transition function
- $q_0 \in Q$ is the initial state
- $Z_0 \in \Gamma$ is the initial stack symbol
- $F \subseteq Q$ is the accepting states

PDA recognizing $\{a^n b^n | n \in \mathbb{N}\}$

$P_1 = (Q = \{q_0, q_1, q_2\}, \Sigma = \{a, b\}, 
\Gamma = \{\text{\#}\}, \delta, \text{\#}, Z_0 = \text{\#}, F = \{q_2\})$

where

\[
\delta(q_0, a, \#) = \{(q_0, a\#)\} \\
\delta(q_0, \epsilon, \#) = \{(q_2, \#)\} \\
\delta(q_0, a, a) = \{(q_0, aa)\} \\
\delta(q_0, b, a) = \{(q_1, \epsilon)\} \\
\delta(q_1, b, a) = \{(q_1, \epsilon)\} \\
\delta(q_1, \epsilon, \#) = \{(q_1, \#)\} \\
\delta(q, \epsilon, \#) = \emptyset \text{ everywhere else}
\]

Example

Consider PDA $P_1$ again on $aabb$:

\[
\begin{align*}
(q_0, aabb, \#) \vdash & (q_0, aabb, a\#) \quad \text{as} \ (q_0, a\#) \in \delta(q_0, a, \#) \\
\vdash & (q_0, bb, aa\#) \quad \text{as} \ (q_0, aa) \in \delta(q_0, a, a) \\
\vdash & (q_1, b, a\#) \quad \text{as} \ (q_1, \epsilon) \in \delta(q_0, b, a) \\
\vdash & (q_1, \epsilon, \#) \quad \text{as} \ (q_1, \epsilon) \in \delta(q_1, b, a) \\
\vdash & (q_2, \#) \quad \text{as} \ (q_2, \#) \in \delta(q_1, \#) \\
\end{align*}
\]

showing that $P_1$ accepts $aabb$ by final state as $q_2 \in F$ and all input consumed.

The Language of a PDA (1)

Two “flavours” of PDAs. **Acceptance by final state:**

$L(P) = \{w | (q_0, w, Z_0) \vdash^* (q, \epsilon, \gamma) \wedge q \in F\}$

**Acceptance by empty stack:**

$L(P) = \{w | (q_0, w, Z_0) \vdash^* (q, \epsilon, \epsilon)\}$

($F$ plays no role and can be left out from the definition of $P$.)

The Language of a PDA (2)

A PDA that accepts by final state can be converted to an equivalent PDA that accepts by empty stack and vice versa.

Both types of PDAs thus describe the same class of languages, the **Context-Free Languages** (CFLs).
Theorem: For a language \( L \subseteq \Sigma^* \),
\[ L = L(G) \text{ for a CFG } G \text{ iff } L = L(P) \text{ for a PDA } P. \]
I.e., the CFGs and the PDAs describe the same class of languages.

Proof: By constructing a PDA \( P \) from a CFG \( G \) and vice versa such that \( L(P) = L(G) \).

We will look at constructing a PDA from a CFG.

### Translating a CFG into a PDA

Given CFG \( G = (N, T, P, S) \),
\[ P(G) = (\{q_0\}, \Sigma = T, \Gamma = N \cup T, \delta, q_0, Z_0 = S) \]
where
\[ \delta(q_0, \epsilon, A) = \{(q_0, \alpha) | A \rightarrow \alpha \in P\} \]
\[ \delta(q_0, a, a) = \{(q_0, \epsilon)\} \text{ for all } a \in T \]
\[ \delta(q_0, w, \gamma) = \emptyset \text{ everywhere else} \]

Acceptance by empty stack.

Note: Highly non-deterministic!

### Deterministic PDAs (DPDAs) (1)

A DPDA is a PDA that has no choice:
A PDA \( P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \) is deterministic iff
\[ |\delta(q, x, z)| + |\delta(q, \epsilon, z)| \leq 1 \text{ for all } q \in Q, x \in \Sigma, z \in \Gamma. \]

Example: \( P_2 \) is not a DPDA.
E.g. \[ |\delta(q_0, 0, A)| + |\delta(q_0, \epsilon, A)| = 0 + 3 \not\leq 1 \]

### Deterministic PDAs (DPDAs) (2)

DPDAs important because can be implemented efficiently. (See lectures on predictive recursive descent parsing.)
But unfortunately:

Theorem: The set of languages accepted by the DPDAs is a strict subset of the languages accepted by PDAs:
\[ L(DPDA) \subset L(PDA) = CFL. \]

However, most context-free languages of practical importance can be described by DPDAs.