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
- $\delta \in Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow P_{\text{fin}}(q \times \Gamma^*)$ 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 recognising \( \{a^n b^n \mid n \in \mathbb{N}\} \)

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

where

\[
\delta(q_0, a, \#) = \{(q_0, a\#)\}
\]
\[
\delta(q_0, \epsilon, \#) = \{(q_2, \#)\}
\]
\[
\delta(q_0, a, a) = \{(q_0, aa)\}
\]
\[
\delta(q_0, b, a) = \{(q_1, \epsilon)\}
\]
\[
\delta(q_1, b, a) = \{(q_1, \epsilon)\}
\]
\[
\delta(q_1, \epsilon, \#) = \{(q_2, \#)\}
\]
\[
\delta(q, w, x) = \emptyset \text{ everywhere else}
\]
An Instantaneous Description (ID)

\((q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*\)

describes the \textit{state} of a PDA computation.
Relations on IDs

\[ \vdash_P P \subseteq ID \times ID \] Read:

\[ id_1 \vdash_P id_2 \]

“PDA \( P \) can move in one step from \( id_1 \) to \( id_2 \).”

1. \( (q, xw, z\gamma) \vdash_P (q', w, \alpha\gamma) \) if \( (q', \alpha) \in \delta(q, x, z) \)

2. \( (q, w, z\gamma) \vdash_P (q', w, \alpha\gamma) \) if \( (q', \alpha) \in \delta(q, \epsilon, z) \)

where \( q, q' \in Q, x \in \Sigma, w \in \Sigma^*, z \in \Gamma, \alpha, \gamma \in \Gamma^* \)
Example

Consider PDA $P_1$ again on $aabb$:

$$(q_0, aabb, \#) \vdash_{P_1} (q_0, abb, a\#) \quad \text{as } (q_0, a\#) \in \delta(q_0, a, \#)$$

$$\vdash_{P_1} (q_0, bb, aa\#) \quad \text{as } (q_0, aa) \in \delta(q_0, a, a)$$

$$\vdash_{P_1} (q_1, b, a\#) \quad \text{as } (q_1, \epsilon) \in \delta(q_0, b, a)$$

$$\vdash_{P_1} (q_1, \epsilon, \#) \quad \text{as } (q_1, \epsilon) \in \delta(q_1, b, a)$$

$$\vdash_{P_1} (q_2, \epsilon, \#) \quad \text{as } (q_2, \#) \in \delta(q_1, \epsilon, \#)$$

showing that $P_1$ accepts $aabb$ by final state as $q_2 \in F$ and all input consumed.
Relations on IDs (cont.)

\( \vdash \subseteq ID \times ID \): The reflexive, transitive closure of \( \vdash \).

Read:

\( p \vdash^* \) $id_1$ $\vdash_P$ $id_2$

“PDA $P$ can move from $id_1$ to $id_2$ in 0 or more steps.”
Relations on IDs (cont.)

\[ \vdash \subseteq ID \times ID : \text{The reflexive, transitive closure of } \vdash. \]

Read:

\[ id_1 \vdash_P \vdash_P id_2 \]

“PDA \( P \) can move from \( id_1 \) to \( id_2 \) in 0 or more steps.”

Examples:

\[ (q_0, aabb, \#) \vdash_{P_1}^* (q_2, \epsilon, \#) \]
Relations on IDs (cont.)

\[ \vdash \subseteq ID \times ID : \text{The reflexive, transitive closure of}\ \vdash. \]

Read:

\[ \vdash \]

"PDA \ P \text{ can move from } id_1 \text{ to } id_2 \text{ in 0 or more steps.}"

Examples:

\[ (q_0, aabb, \#) \vdash_{P_1} (q_2, \epsilon, \#) \]

For any PDA \ P and ID \ id: \ id \vdash_P \ id
Two “flavours” of PDAs. **Acceptance by final state:**

\[ L(P) = \{ w \mid (q_0, w, Z_0) \xrightarrow{\ast}_P (q, \epsilon, \gamma) \land q \in F \} \]
The Language of a PDA (1)

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

\[ L(P) = \{w \mid (q_0, w, Z_0) \xrightarrow{}^* (q, \epsilon, \gamma) \land q \in F\} \]

**Acceptance by empty stack:**

\[ L(P) = \{w \mid (q_0, w, Z_0) \xrightarrow{}^* (q, \epsilon, \epsilon)\} \]

(* plays no role and can be left out from the definition of \( P \).)
A PDA that accepts by final state can be converted to an equivalent PDA that accepts by empty stack and vice versa.
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).
PDAs and CFGs

Theorem: For a language \( L \subseteq \Sigma^* \),
\[
L = L(G) \text{ for a CFG } G \iff L = L(P) \text{ for a PDA } P.
\]
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

$$\delta(q_0, \epsilon, A) = \{(q_0, \alpha) \mid A \rightarrow \alpha \in P\}$$

$$\delta(q_0, a, a) = \{(q_0, \epsilon)\} \text{ for all } a \in T$$

$$\delta(q_0, w, \gamma) = \emptyset \text{ everywhere else}$$

Acceptance by empty stack.
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

$$\delta(q_0, \epsilon, A) = \{(q_0, \alpha) \mid A \to \alpha \in P\}$$
$$\delta(q_0, a, a) = \{(q_0, \epsilon)\} \text{ for all } a \in T$$
$$\delta(q_0, w, \gamma) = \emptyset \text{ everywhere else}$$

Acceptance by empty stack.

Note: Highly non-deterministic!
Example: Translating a CFG into a PDA

Consider the grammar $G_2$:

$$A \rightarrow 0A0 \mid 1A1 \mid \epsilon$$
Example: Translating a CFG into a PDA

Consider the grammar $G_2$:

$$A \rightarrow 0A0 \mid 1A1 \mid \epsilon$$

Contract PDA $P_2 = P(G_2)$. On white board.
A DPDA is a PDA that has no choice:

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

$$|\delta(q, x, z)| + |\delta(q, \epsilon, z)| \leq 1 \text{ for all } q \in Q, x \in \Sigma, z \in \Gamma.$$
A DPDA is a PDA that has \textit{no} choice:

A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is \textit{deterministic} iff

$|\delta(q, x, z)| + |\delta(q, \epsilon, z)| \leq 1$ 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.)
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 \textit{strict subset} of the languages accepted by PDAs: \( L(\text{DPDA}) \subset L(\text{PDA}) = \text{CFL} \).
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.