Pushdown Automata

- like NFA-$\Lambda$ but also has a stack

- transition takes the current state, the current input symbol, and the top-of-the-stack symbol (which is removed from the stack) and returns a state and a string to place in the stack

- the automaton starts in the initial state, reading the first symbol on the tape, and having a special initial symbol on the stack

- automaton accepts a string $x$ if, after reading all of $x$, it can get to an accepting state (the stack content does not matter)

- if the stack is empty, the automaton is stuck
Pushdown Automata

Example: \( L = \{ a^k b^k \mid k \geq 0 \} \)

\[ \text{push \( a \)'s onto stack} \]

\[ q_{\text{init}} \]

\[ b, a / \Lambda \]

\[ \Lambda, a | a \rightarrow \text{pop \( a \)'s, check \( b \)'s} \]

\[ q_0 \]

\[ \Lambda, Z_0 | Z_0 \rightarrow \text{accept} \]

\[ \Lambda, a, Z_0 | Z_0 \rightarrow a, a, Z_0 \]

\[ a, a | a \rightarrow b, a, Z_0 \]

\[ b, b, a | a \rightarrow b, a, Z_0 \]

\[ b, b, b, Z_0 \rightarrow b, b, b, Z_0 \]

See what happens with \( \text{aaa bbb b} \) (doesn't accept)
Example: \( L = \{ w \in \{a, b\}^* \mid w \text{ is an even-length palindrome} \} = \{ uu^r \mid u \in \{a, b\}^* \} \)
Pushdown Automata

Def: A **pushdown automaton** (PDA) is a 7-tuple $(Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ where

- $Q$ is a finite set of states
- $\Sigma$ is a finite input alphabet
- $\Gamma$ is a finite stack alphabet
- $q_0 \in Q$ is the initial state
- $Z_0 \in \Gamma$ is the initial stack symbol
- $A \subseteq Q$ is the set of accepting states
- $\delta$ is the transition function from $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma$ to finite subsets of $Q \times \Gamma^*$
Def: Let $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ be a PDA. A configuration is a triple $(p, x, \alpha)$ where $p$ is the current state, $x$ is the rest of the input (the yet-unread part), and $\alpha$ is the content of the stack (the top is on the left).

We write $(p, x, \alpha) \vdash_M (q, y, \beta)$ if we can get from the configuration $(p, x, \alpha)$ to the configuration $(q, y, \beta)$ using one transition of $M$.

Similarly we write $(p, x, \alpha) \vdash_M^* (q, y, \beta)$ if we can get from $(p, x, \alpha)$ to $(q, y, \beta)$ through a finite sequence of transitions of $M$.

The language accepted by $M$:

$$L(M) = \{ x \in \Sigma^* | (q_0, x, Z_0) \vdash_M^* (p, \Lambda, \alpha) \text{ for some } p \in A \}$$
Def: Let $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ be a PDA. $M$ is deterministic (DPDA) if

$$\cup \{ \Lambda \}$$

for every $q \in Q$, for every $\sigma \in \Sigma$ and every $\tau \in \Gamma$:

$$| \delta(q, \sigma, \tau) | \leq 1$$

if there exists a $\Lambda$-transition from a state $q \in Q$ with top of the stack $\tau$ then no $\Sigma$-transitions from $q \in Q$ & top of the stack $\tau$ are allowed

$$\left( \text{if for } q \in Q \text{, } \tau \in \Gamma \text{ s.t. } \delta(q, \Lambda, \tau) \neq \emptyset \text{, then} \right.$$  

$$\forall \sigma \in \Sigma : \delta(q, \sigma, \tau) = \emptyset \left. \right)$$

A language $L$ is a deterministic context-free language (DCFL) if there exists a DPDA accepting $L$. 
Deterministic Pushdown Automata

Example: Give a DPDA for \( L = \{ a^k b^k | k \geq 1 \} \).

Can you give a DPDA for \( L_2 = \{ a^k b^k | k \geq 0 \} \)?
Determinism vs. Nondeterminism

For finite automata:

\[ \text{FA} \quad \& \quad \text{NFA} \quad \& \quad \text{NFA-\Delta} \quad \text{all accept the same class of languages (regular)} \]

For PDA:

\[ \text{NPDA} = \text{PDA} \quad \text{accept more langs. than DPDA!} \]

There exists a language which can be accepted by a PDA but not by a DPDA!

\[ \{ww^r \mid w \in \{a,b\}^*\} \quad \text{can be accepted by a PDA} \leftarrow \text{here done} \]

\[ \text{cannot be accepted by a DPDA!} \leftarrow \text{in the book} \]

Regular \( \not\subseteq \) DPDA \( \not\subseteq \) PDA
Example:

Is the language given by $S \rightarrow SS \mid (S) \mid \Lambda$ a DCFL?

- \hspace{2cm} (\text{well-parenthesized strings})

- \hspace{2cm} \Lambda, () \rightarrow ()(), ()

- \hspace{2cm} \text{def. CFL} \leftarrow \text{def: L is DCFL is 3 DPDA for L.}
Determinism vs. Nondeterminism

Example: \( \{ x \in \{a,b\}^* \mid \text{number of } a\text{'s in } x = \text{number of } b\text{'s in } x \} \)

TRY: give a DPDA for this \( L \)

PDA make a DPDA as before
CFG vs. PDA

It is possible to show that every language generated by a CFG can be accepted by a PDA and vice versa.

**Thm:** Let $G = (V, \Sigma_G, S, P)$ be a CFG. Then there exists a PDA $M = (Q, \Sigma_M, \Gamma, q_0, Z_0, A, \delta)$ such that $L(M) = L(G)$.

**Thm:** Let $M = (Q, \Sigma_M, \Gamma, q_0, Z_0, A, \delta)$ be a PDA. Then there exists a CFG $G = (V, \Sigma_G, S, P)$ such that $L(G) = L(M)$.

Which one is easier to prove?
**CFG vs. PDA**

**Thm:** Let $G = (V, \Sigma_G, S, P)$ be a CFG. Then there exists a PDA $M = (Q, \Sigma_M, \Gamma, q_0, Z_0, A, \delta)$ such that $L(M) = L(G)$.

**Example:**

$S \rightarrow S + T | T$

$T \rightarrow T * R | R$

$R \rightarrow (S) | x$

$S \Rightarrow S+T \Rightarrow T+T \Rightarrow R+T \Rightarrow x+T \Rightarrow x+T*R \Rightarrow \ldots$