Chomsky Normal Form

A CFG is in Chomsky Normal Form (CNF) if every rule is in one of the following three forms:

- \( S \rightarrow \epsilon \)
- \( A \rightarrow B \ C \) \( B, C \) are variables \( \neq S \)
- \( A \rightarrow a \) \( a \) is a terminal

(S is the start variable; \( A \) is any variable, including \( S \))

**Theorem:** Any CFG can be converted into an equivalent CFG (generating the same CFL) in Chomsky Normal Form
(proof done on the board – read Sipser Thm 2.9)
Finite Automaton

Pushdown Automaton
Informal Definition of **Acceptance**

- A pushdown automation accepts if, after reading the entire input, it ends in an accept state
  - Sometimes: (with an empty stack)
Definition: A (non-deterministic) PDA is a tuple $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$, where:

- $Q$ is a finite set of states
- $\Sigma$ is the input alphabet
- $\Gamma$ is the stack alphabet
- $\delta : Q \times \Sigma \times \Gamma \rightarrow 2^{Q \times \Gamma}$ (non-determinism)
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states

$2^S$ is the set of subsets of $S$ 
$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$, $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$

Formal Definition of Acceptance

PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts a word $w \in \Sigma^*$ where $w = w_1w_2w_3\ldots w_m$ with $w_i \in \Sigma_\epsilon$ if there exists a sequence 

$$(q_0, s_0) \rightarrow (q_1, s_1) \rightarrow (q_2, s_2) \rightarrow \ldots \rightarrow (q_m, s_m)$$

where

- $s_i \in \Gamma^*$ (represent the stack), with $s_0 = \epsilon$, $q_m \in F$,
- $(q_{i+1}, b) \in \delta(q_i, w_{i+1}, a)$
  where $s_i = at$ and $s_{i+1} = bt$, $a,b \in \Gamma_\epsilon$, $t \in \Gamma^*$
EVEN-LENGTH PALINDROMES

\[ \Sigma = \{a, b, c, ..., z\}, \sigma \in \Sigma \]
Build a PDA to recognize
\[ L = \{ a^i b^j c^k \mid i, j, k \geq 0 \text{ and } (i = j \text{ or } i = k) \} \]

Theorem

A Language is generated by a CFG \iff It is recognized by a PDA
Theorem

A Language is generated by a CFG

⇒

It is recognized by a PDA

Suppose L is generated by a CFG G = (V, Σ, R, S)
Construct P = (Q, Σ, Γ, δ, q, F) that recognizes L

Intuition (warning: not a formal proof!)

Map a derivation in the CFG G to an accepting sequence for the PDA P

Let w ∈ L(G)
There is a derivation in G:
S ⇒ α₁ ⇒ α₂ ⇒ ... ⇒ αₘ₋₁ ⇒ w where αᵢ ∈ (Σ ∪ V)*

We map it to an accepting sequence
(q₀, s₀) → (q₁, s₁) → ... (q₂, s₂) → ... → (qₘ, sₘ) → (qᵢ, sᵢ)
where
q₁ = q₂ = ... = qₘ = q_loop, qᵢ ∈ F,
s₀ = S$, s₁ = α₁$, (1 ≤ i ≤ m)
sᵢ is obtained from sᵢ₋₁ (1 ≤ i ≤ m) by using substitution at corresponding step of the derivation and matching terminals on the top of the stack with the input

(see picture on slide 16)
Suppose $L$ is generated by a CFG $G = (V, \Sigma, R, S)$.
Construct $P = (Q, \Sigma, \Gamma, \delta, q, F)$ that recognizes $L$.

1. Place the marker symbol $\$$ and the start variable $S$ on the stack.
2. Repeat the following steps forever:
   a. If top of stack is a variable, nondeterministically select a rule that matches the variable and push the result into the stack.
   b. If top of stack is a terminal, read the next symbol from input and compare it to that terminal. If different, reject.
   c. If top of stack is $\$$, then enter accept state. Accept if the input has all been read.

Note: RHS is a string (non-std notation just for intuition)
A Language is generated by a CFG
⇒
It is recognized by a PDA
Next Steps

- Read Sipser 2.3 in preparation for next lecture