Lecture 8:

• Equivalence between PDAs and CFGs
• Closure Properties

Reading:
Sipser Ch 2.2
Pushdown Automaton (the idea)

- **Nondeterministic** finite automaton + stack
- Stack has unlimited size, but machine can only manipulate (push, pop, read) symbol at the top

Input: 

\[
\begin{array}{cccc}
\text{a} & \text{b} & \text{a} & \text{a} & \ldots \\
\end{array}
\]

Transitions of the form:

\[
p \xrightarrow{a, x} x' = \text{push} \quad q
\]
Example: Even Palindromes

\[
\begin{align*}
q_0 & \xrightarrow{\epsilon, \epsilon} p \\
p & \xrightarrow{a, \epsilon} a \\
p & \xrightarrow{b, \epsilon} b \\
q_f & \xrightarrow{\epsilon, \epsilon} q \\
q & \xrightarrow{a, a} \epsilon \\
q & \xrightarrow{b, b} \epsilon
\end{align*}
\]
Pushdown Automaton (formal)

A PDA is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

- $Q$ is a finite set of states
- $\Sigma$ is the input alphabet
- $\Gamma$ is the stack alphabet
- $\delta : Q \times \Sigma \varepsilon \times \Gamma \varepsilon \rightarrow P(Q \times \Gamma \varepsilon)$ is the transition function
- $q_0$ is the start state
- $F$ is the set of final states

$M$ accepts a string $w$ if, starting from $q_0$ and an empty stack, there exists a path to an accept state that can be followed by reading all of $w$. 
Example

\[ L = \{ w | w \text{ has an equal number of } a \text{'s and } b \text{'s} \} \]

Algorithmic description

1. Push \$ to stack

2. Repeat indefinitely:
   - Do one of the following:
     - Read character \& push to the stack
     - Match next character to stack \& pop it

Accept if \$ on the stack
Example

$L = \{w | w \text{ has an equal number of } a\text{'s and } b\text{'s}\}$

State diagram
CFGs vs. PDAs

The language $L(M)$ of a PDA $M$ is the set of all strings it accepts.

Theorem: The class of languages recognized by PDAs is exactly the context-free languages.

**Theorem 1:** Every CFG has an equivalent PDA
**Theorem 2:** Every PDA has an equivalent CFG
CFG -> PDA
CFG -> PDA Conversion

Suppose language $L$ is generated by CFG $G = (V, \Sigma, R, S)$

**Goal:** Construct a PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ recognizing $L$

**Idea:** $M$ will guess the steps of the CFG derivation of its input $w$, and use its stack to check the derivation.

A helpful intermediate abstraction

**Generalized PDA:** Can push an entire string to the stack in one move.
Algorithmic Description

1. Place $\$ \text{ and the start variable } S \text{ on the stack} \quad \text{(Stack: } S \$\text{)}
2. Repeat forever:
   a) If the top of the stack holds variable $A$: 
      Nondeterministically guess a rule $(A \rightarrow u) \in R$
      Replace $A$ with $u$ on the stack
   b) If the top of the stack holds terminal $\sigma$: 
      Pop $\sigma$ and verify that it matches the next input char
   c) If the top of the stack holds $\$":
      Accept if the input is empty
\[ \varepsilon, \varepsilon \rightarrow S\$

\[ q_0 \]

\[ \varepsilon, S \rightarrow \varepsilon \]

\[ q_{\text{loop}} \]

\[ \varepsilon, A \rightarrow u \quad [\text{for every rule } A \rightarrow u] \]
\[ \sigma, \sigma \rightarrow \varepsilon \quad [\text{for every terminal } A \rightarrow \sigma] \]

\[ q_f \]

\[ \varepsilon, \varepsilon \rightarrow S\$ \]
Example

\[ S \rightarrow aTb \]
\[ T \rightarrow Ta | \varepsilon \]

\[ \varepsilon, \varepsilon \rightarrow S$ \]
\[ \varepsilon, S \rightarrow aTb \]
\[ e, T \rightarrow \varepsilon \]
\[ e, T \rightarrow Ta \]
\[ a, a \rightarrow \varepsilon \]
\[ b, b \rightarrow \varepsilon \]

Input: $aaa b$

Stack:
- $$

Intermediate steps:
- $S \Rightarrow aTb$
- $aTb \Rightarrow aa b$
Example

\[ S \rightarrow aTb \]
\[ T \rightarrow Ta \mid \varepsilon \]
PDA -> CFG
PDA -> CFG Conversion

Theorem 2: Every PDA has an equivalent CFG

Suppose $L$ is recognized by PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

Goal: Construct a CFG $G = (V, \Sigma, R, S)$ generating $L$

First simplify $M$ so that:

1. It has a single accept state $q_f$
2. It empties the stack before accepting
3. Every transition either pushes a symbol or pops a symbol (but not both)
Simplification Example

\[ \begin{align*}
\varepsilon, \varepsilon & \rightarrow \$ \\
\varepsilon, \varepsilon & \rightarrow \varepsilon \\
a, \varepsilon & \rightarrow a \\
b, \varepsilon & \rightarrow b \\
a, a & \rightarrow \varepsilon \\
b, b & \rightarrow \varepsilon
\end{align*} \]
Conversion Idea

Variables: $A_{pq}$ for every pair of states $p, q$ in PDA $M$

Idea: $A_{pq}$ generates all strings that can take $M$ from $p$ (with an empty stack) to $q$ (with an empty stack)

\[ V = \frac{1}{2} \left\{ A_{pq} \mid p, q \in Q \right\} \]

\[ S = A_{q_0 q_{\text{final}}} \]
Example

What strings should $A_{q_0q_1}$ generate?
None

What strings should $A_{q_1q_3}$ generate?
$\exists \: \omega \omega^R \mid \omega \in \{a,b\}^*$

What strings should $A_{q_1q_4}$ generate?
None
What rules should define $A_{pq}$?

Let $x$ be a string generated by $A_{pq}$

Two cases:

1) Stack first empties after reading all of $x$

2) Stack empties before reaching the end of $x$
1. Stack first empties after reading all of $x$

Add rule $A_{pq} \rightarrow aA_{rs}b$
2. Stack empties before reaching the end of $x$

Add rule $A_{pq} \rightarrow A_{pr}A_{rq}$
Formal CFG Construction

\[ V = \{ A_{pq} \mid p, q \in Q \} \]
\[ S = A_{q_0q_f} \]

Three kinds of rules:

1. For every \( p, q, r, s \in Q, \ t \in \Gamma, \ a, b \in \Sigma \) :
   
   If \( (r, t) \in \delta(p, a, \varepsilon) \) and \( (q, \varepsilon) \in \delta(s, b, t) \),
   
   include the rule \( A_{pq} \rightarrow aA_{rs}b \)

2. For every \( p, q, r \in Q \), include the rule \( A_{pq} \rightarrow A_{pr}A_{rq} \)

3. For every \( p \in Q \), include the rule \( A_{pp} \rightarrow \varepsilon \)
Sketch of proof that CFG generates $L(M)$

Claim: $A_{pq} \Rightarrow^* x$ if and only if $x$ takes $M$ from $p$ to $q$, beginning and ending with empty stack

Proof idea:

$\Rightarrow$ Induction on number of steps of derivation of $x$ from $A_{pq}$

$\Leftarrow$ Induction on number of steps of computation taking $M$ from $p$ to $q$
Closure Properties
Closure Properties

• The class of CFLs is closed under
  Union
  Concatenation
  Star
  Intersection with regular languages

  \[ A \text{ context-free, } B \text{ regular } \Rightarrow A \cap B \text{ CFL} \]

• Beware: It is not closed under intersection or complement
  (Find counterexamples!)

  \[ A \text{ CFL, } B \text{ CFL, } A \cap B \text{ not CFL} \]
Closure under union (Proof 1)

Let $A$ be a CFL recognized by PDA $M_A$ and let $B$ be a CFL recognized by PDA $M_B$

**Goal:** Construct a PDA recognizing $A \cup B$
Closure under union (Proof 2)

Let \( A \) be a CFL generated by CFG \( G_A \) and let \( B \) be a CFL recognized by CFG \( G_B \)

**Goal:** Construct a CFG \( G \) recognizing \( A \cup B \)

\[
G_A = (V_A, \Sigma_A, R_A, S_A) \\
G_B = (V_B, \Sigma_B, R_B, S_B)
\]

Relabel variables so \( V_A \) and \( V_B \) are disjoint

Let \( G = (V, \Sigma, R, S) \)

\[
R = R_A \cup R_B \cup \{S \to S_A, S \to S_B\}
\]