Last time:
• Pumping lemma for CFLs
• Proving that a language is not CF

Today:
• Equivalence of CFGs and PDAs
Read on your own

1. Ambiguous CFGs.
2. Chomsky normal form for CFGs
3. (Skipping Chapter 2.4 in Sipser).
The language of P is the set of strings it accepts. PDAs are **nondeterministic**.
When PDA takes a nondeterministic step, what happens to the stack?

A. First 0 is pushed onto the stack, then 1.
B. First 1 is pushed onto the stack, then 0.
C. Now PDA has access to two stacks and it pushes 0 onto one and 1 onto the other.
D. Two different computational branches are available to the PDA: it pushes exactly one symbol (0 or 1) onto the stack on each branch.
E. None of the above
A language is generated by a CFG

\[ \iff \]

It is recognized by a PDA
Suppose $L$ is generated by a CFG $G = (V, \Sigma, R, S)$. Construct a PDA $P = (Q, \Sigma, \Gamma, \delta, q, F)$ that recognizes $L$.

**Idea:** $P$ will guess steps of a derivation of its input $w$ and use its stack to derive it.
(1) Place the marker symbol $ and the start variable on the stack.

(2) Repeat forever:

(a) If a variable $V$ is on top of the stack, and $(V \rightarrow s) \in R$, pop $V$ and push string $s$ on the stack in reverse order.

(b) If a terminal is on top of the stack, pop it and match it with input.

(3) On $(\varepsilon, \$)$, accept.
(q\textsubscript{start}) Push S$ and go to q\textsubscript{loop}

(q\textsubscript{loop}) Repeat the following steps forever:
  (a) On (ε,V) where (V → s) ∈ R, push s and go to q\textsubscript{loop}
  (b) On (σ,σ), pop σ and go to q\textsubscript{loop}
  (c) On (ε,$) go to q\textsubscript{accept}

Otherwise, the PDA will get stuck!
Designing PDA

- $\epsilon, \epsilon \rightarrow S\$$
- $\epsilon, A \rightarrow w$ for each rule $A \rightarrow w$
- $a, a \rightarrow \epsilon$ for each terminal $a$
- $\epsilon, \$ \rightarrow \epsilon$
S → aTb
T → Ta | ε
A language is generated by a CFG

It is recognized by a PDA
Converting a PDA to a CFG

Given PDA $P = (Q, \Sigma, \Gamma, \delta, q, F)$
Construct a CFG $G = (V, \Sigma, R, S)$ with $L(G) = L(P)$

First, **simplify $P$** so that:

1. It has a single accept state, $q_{\text{accept}}$
2. It empties the stack before accepting
3. Each transition does exactly one of:
   - pushes a symbol;
   - pops a symbol.
SIMPLIFY

$\varepsilon, \varepsilon \rightarrow \$\n
$\varepsilon, \varepsilon \rightarrow \emptyset$

$\varepsilon, \varepsilon \rightarrow \emptyset$

$1,0 \rightarrow \varepsilon$

$1,0 \rightarrow \varepsilon$

$\varepsilon, \varepsilon \rightarrow \emptyset$

$\varepsilon, \varepsilon \rightarrow \emptyset$

$\varepsilon, \varepsilon \rightarrow \emptyset$

$\varepsilon, \varepsilon \rightarrow \emptyset$

$\varepsilon, \varepsilon \rightarrow \emptyset$

$\varepsilon, \varepsilon \rightarrow \emptyset$

$\varepsilon, \varepsilon \rightarrow \emptyset$
For each pair of states p and q in P, add a variable $A_{pq}$ to CFG that generates all strings that can take P from p to q without changing the stack*

$$V = \{ A_{pq} \mid p,q \in Q \}$$

$$S = A_{q_0q_{\text{accept}}}$$

*Starting from any stack S in p, including empty stack, P has stack S at q.
What strings does $A_{q_0q_1}$ generate? none

What strings does $A_{q_1q_2}$ generate? $\{0^n1^n \mid n > 0\}$

What strings does $A_{q_1q_3}$ generate? none
Apq generates all strings that take P from p to q without changing the stack

Let x be such a string

• P’s first move on x must be a push
• P’s last move on x must be a pop

Consider the stack while reading x. Either:

1. New portion of the stack first empties only at the end of x
2. New portion empties before the end of x
1. New portion of the stack first empties only at the end of x

\[ A_{pq} \rightarrow aA_{rs}b \]
2. New portion empties before the end of $x$
$V = \{ A_{pq} | p, q \in Q \}$

$S = A_{q_0 q_{accept}}$

For each $p, q, r, s \in Q$, $t \in \Gamma$ and $a, b \in \Sigma_\epsilon$

If $(r, t) \in \delta(p, a, \epsilon)$ and $(q, \epsilon) \in \delta(s, b, t)$

Then add the rule $A_{pq} \rightarrow aA_{rs}b$

For each $p, q, r \in Q$,

add the rule $A_{pq} \rightarrow A_{pr}A_{rq}$

For each $p \in Q$,

add the rule $A_{pp} \rightarrow \epsilon$
What strings does $A_{q_0 q_1}$ generate? none
What strings does $A_{q_1 q_2}$ generate? $\{0^n1^n \mid n > 0\}$
What strings does $A_{q_1 q_3}$ generate? none
A language is generated by a CFG

\[ \iff \]

It is recognized by a PDA