What’s on our plate today?

✦ Cliff’s notes for equivalence of CFLs and L(PDAs)
  ➤ L is a CFL ⇒ L = L(M) for some PDA M
  ➤ L = L(M) for some PDA M ⇒ L = L(G) for some CFG G

✦ Pumping Lemma (one last time)
  ➤ Statement of Pumping Lemma for CFLs
  ➤ Proof: See class notes from last time and textbook
  ➤ Application: Showing a given L is not a CFL

✦ Introduction to Turing Machines

From CFLs to PDAs

✦ L is a CFL ⇒ L = L(M) for some PDA M

✦ Proof Summary:
  ➤ L is a CFL means L = L(G) for some CFG G = (V, Σ, R, S)
  ➤ Construct PDA M = (Q, Σ, Γ, δ, q_0, {q_{acc}})
    M has only 4 main states (plus a few more for pushing strings)
    Q = {q_0, q_1, q_2, q_{acc}} \cup E where E are states used in 2 below
  ➤ δ has 4 components:
    1. Init. Stack: δ(q_0, ε, ε) = {(q_1, $)} and δ(q_1, ε, ε) = {(q_2, S)}
    2. Push & generate strings: δ(q_2, ε, A) = {(q_2, w)} for A → w in R
    3. Pop & match to input: δ(q_2, a, a) = {(q_2, ε)}
    4. Accept if stack empty: δ(q_2, ε, $) = {(q_{acc}, ε)}

✦ Can prove by induction: w ∈ L iff w ∈ L(M)
From PDAs to CFLs

- \( L = L(M) \) for some PDA \( M \) \( \Rightarrow \) \( L = L(G) \) for some CFG \( G \)
- Proof Summary: Simulate \( M \)'s computation using a CFG
  - First, simplify \( M \): 1. Only 1 accept state, 2. \( M \) empties stack before accepting, 3. Each transition either Push or Pop, not both or neither. Let \( M = (Q, \Sigma, \Gamma, \delta, q_0, \{ q_{acc} \}) \)
  - Construct grammar \( G = (V, \Sigma, R, S) \)
  - Basic Idea: Define variables \( A_{pq} \) for simulating \( M \)
  - \( A_{pq} \) generates all strings \( w \) such that \( w \) takes \( M \) from state \( p \) with empty stack to state \( q \) with empty stack
  - Then, \( A_{q0qacc} \) generates all strings \( w \) accepted by \( M \)

From PDAs to CFLs (cont.)

- \( L = L(M) \) for some PDA \( M \) \( \Rightarrow \) \( L = L(G) \) for some CFG \( G \)
- Proof (cont.)
  - Construct grammar \( G = (V, \Sigma, R, S) \) where
    \[
    V = \{ A_{pq} | p, q \in Q \} \\
    S = A_{q0qacc} \\
    R = \{ A_{pq} \rightarrow aA_{rs}b | p, q \in Q \} \\
    \cup \{ A_{pq} \rightarrow A_{ps}A_{rq} | p, q, r \in Q \} \\
    \cup \{ A_{qq} \rightarrow \epsilon | q \in Q \} \\
    \]
  - See text for proof by induction: \( w \in L(M) \) iff \( w \in L(G) \)
  - Try to get \( G \) from \( M \) where \( L(M) = \{ 0^n1^n | n \geq 1 \} \)
Pumping Lemma for CFLs

✦ Intuition: If \( L \) is CF, then some CFG \( G \) produces strings in \( L \)
  ➤ If some string in \( L \) is very long, it will have a very tall parse tree
  ➤ If a parse tree is taller than the number of distinct variables in \( G \),
    then some variable \( A \) repeats ⇒ \( A \) will have at least two sub-trees
  ➤ We can pump up the original string by replacing \( A \)’s smaller sub-
    tree with larger, and pump down by replacing larger with smaller

✦ Pumping Lemma for CFLs in all its glory:
  ➤ If \( L \) is a CFL, then there is a number \( p \) (the “pumping length”) such that
    for all strings \( s \) in \( L \) such that \( |s| \geq p \), there exist \( u, v, x, y, \) and \( z \) such that
    \( s = uvxyz \) and:
    1. \( uv^ixyz \in L \) for all \( i \geq 0 \), and
    2. \( |vy| \geq 1 \), and
    3. \( |vxy| \leq p \).

R. Rao, CSE 322

Why is the PL useful?

✦ Can use the pumping lemma to show a language \( L \) is not context-free
  ➤ 5 steps for a proof by contradiction:
    1. Assume \( L \) is a CFL.
    2. Let \( p \) be the pumping length for \( L \) given by the pumping
      lemma for CFLs.
    3. Choose cleverly an \( s \) in \( L \) of length at least \( p \), such that
    4. For all possible ways of decomposing \( s \) into \( uvxyz \),
      where \( |vy| \geq 1 \) and \( |vxy| \leq p \),
    5. We can choose an \( i \geq 0 \) such that \( uv^ixy^iz \) is not in \( L \).

✦ In-Class Examples: Show the following are not CFLs
  ➤ \( L = \{0^n1^n0^n \mid n \geq 0 \} \) and \( L = \{0^n \mid n \) is a prime number\}
Using the Pumping Lemma

✦ Show $L = \{0^n \mid n \text{ is a prime number}\}$ is not a CFL
  1. Assume $L$ is a CFL.
  2. Let $p$ be the pumping length for $L$ given by the pumping lemma for CFLs.
  3. Let $s = 0^n$ where $n$ is a prime $\geq p$
  4. Consider all possible ways of decomposing $s$ into $uvxyz$, where $|vy| \geq 1$ and $|vxy| \leq p$.

Then, $vy = 0^r$ and $uxz = 0^q$ where $r + q = n$ and $r \geq 1$

5. We need an $i \geq 0$ such that $uv^ixy^iz = 0^{ir+q}$ is not in $L$.

$(i = 0$ won’t work because $q$ could be prime: e.g. $2 + 17 = 19)$
Choose $i = (q + 2r)$. Then, $ir + q = qr + 2r + 2r^2 + q = q(r+1)+2r(r+1) = (q+2r)(r+1) = \text{not prime (since } r \geq 1)$.

So, $0^{ir+q}$ is not in $L \Rightarrow$ contradicts pumping lemma. $L$ is not a CFL.

Two cool results about CFLs

✦ CFLs are not closed under intersection
  ➤ **Proof**: $L_1 = \{0^n1^n0^n \mid n, m \geq 0\}$ and $L_2 = \{0^n1^n0^n \mid n, m \geq 0\}$
  are both CFLs but $L_1 \cap L_2 = \{0^n1^n0^n \mid n \geq 0\}$ is not a CFL.

✦ CFLs are not closed under complementation
  ➤ **Proof by contradiction**:
  Suppose CFLs are closed under complement.

  Then, for $L_1, L_2$ above, $\overline{L_1 \cup L_2 = L_1 \cap \overline{L_2}}$ must be a CFL (since CFLs are closed under $\cup$ -- see homework #5, problem 1).

  But, $\overline{L_1 \cup L_2 = L_1 \cap L_2}$ (by de Morgan’s law).

  $L_1 \cap \overline{L_2} = \{0^n1^n0^n \mid n \geq 0\}$ is not a CFL $\Rightarrow$ contradiction.

  Therefore CFLs are not closed under complementation.
Can we make PDAs more powerful?

✦ PDA = NFA +

✦ What if we allow arbitrary reads/writes to the stack instead of only push and pop?

Enter….Turing Machines!

Just like a DFA except:
- You have an infinite “tape” memory (or scratchpad) on which you receive your input and on which you can do your calculations
- You can read 1 symbol at a time from a cell on the tape, write 1 symbol, then move the read/write pointer (head) left (L) or right (R)
Who’s Turing?

✦ Alan Turing (1912-1954): one of the most brilliant mathematicians of the 20th century (one of the founding “fathers” of computing)

✦ Click on “Theory Hall of Fame” link on class web under “Lectures”

✦ Introduced the Turing machine as a formal model of what it means to compute and solve a problem (i.e. an “algorithm”)


How do Turing Machines compute?

✦ \( \delta(q_0, 1) = (q_1, 0, R) \) (R = right, L = left)

✦ In terms of “Configurations”: 11q_0110 \( \Rightarrow \) 110q_110
Solving Problems with Turing Machines

✦ We know $L = \{0^n1^n0^n \mid n \geq 0\}$ cannot be accepted by any PDA
✦ Design a Turing Machine (TM) that accepts $L$
  ✔ Write down its operation in words…

Next Time…

✦ Formal definition of TMs
✦ Solving problems with TMs
✦ Varieties of TMs (multi-tape, nondeterministic, etc.)
✦ Homework #5 due on Friday!

Can I have my Oscar now?