Lecture 4/7. Technique for proving non-CFL ness of a language $L \subseteq \Sigma^*$:

Let any (sufficiently large) $p > 0$ be given. Take $x = \ldots$ such that $x \in L$ and $|x| \geq p$ indeed often $|x| > 2p$.

Let any breakdown $x = uv^iw^jx$ subject to $|uvw| \leq p$ and $uw \in \Sigma^*$ be given.

Take $i = \ldots$ so that you can get $x^{(i)} = uv^iw^jx \in L$, $i=0$ is allowed.

Then $L$ is not a CFL.

Example: $L = \{a^n b^n c^n : n \geq 0\}$. Let any $p > 0$ be given.

Take $x = a^p b^p c^p$

$$x = \underbrace{aa \ldots} \underbrace{abbb \ldots} \underbrace{ccccc \ldots} = p = p = p$$

Take $x^{(i)} = uv^i w^j x$. By $|uvw| \leq p$, $uw$ can't both contain an 'a' and a 'c', so either $A_q(x^{(i)})$ or $A_c(x^{(i)})$ remains equal to $p$. We get at least one other character, so we cannot have all of $A_q$, $A_b$, $A_c$ equal to $p$, so $x^{(i)}$ can't possibly belong to $L$. Hence $L$ is not a CFL.

\[ \square \]

The same proof works for $L' = \{x \in \{a,b,c\}^* : \#a(x) = \#b(x) = \#c(x) \}$. It is a regular set, if $L'$ were a CFL then so would be $L \cup L'$, but this equals $L$. \[ \square \]
Note \( L = \{ a^n b^n c^r : n, r \geq 0 \} \cap \{ a^n b^n c^r : n, r \geq 0 \} \). The two languages being intersected are CF-Ls.

CFG for \( L_1 = S \rightarrow TC, C \rightarrow cC \mid \varepsilon \) \( \varepsilon \) if \( c^* \).

CFG for \( L_2 = S \rightarrow AT, A \rightarrow aA \mid \varepsilon, T \rightarrow bT' \) \( \varepsilon \).

The class of CF-Ls, unlike the class of regular languages, is not closed under intersection. \( \widehat{L} \) is a CF-L (remarked on hw).

- It is closed under union.
- Hence it is also not closed under complements. If it were, we would get closure under \( \cap \) via \( L_1 \cap L_2 = \cap \) \( L_1 \cup L_2 \).

However, if \( L \) is a CF-L and \( R \) is a regular set, then \( L \cap R \) (and of course \( L \cup R \)) are CF-Ls.

Why the Cartesian product construction doesn't work for N:\no\text{POA}

\[ X \in L_1 \cap L_2 \]

\[ X = x_1 x_2 \ldots x_n \]

NPOA

N \( \text{st.} \)

\( L(N_1) = L \)

NPOA

N \( \text{st.} \)

\( L(N_2) = L_2 \)

If we want one POA \( N \) such that \( L(N) = L(N_1) \cap L(N_2) \), they must agree on sharing an stack.

But if \( N_2 \) is a DFA, or even an NFA, it needs no stack and can patiently react to inputs \( N_1 \) reads.
Example 2: \( A = \sum a^m b^n c^m d^n : m, n \geq 0 \) is not a CFL.

Proof: Let any (sufficiently large) \( p > 0 \) be given. Take \( x = a^p b^p c^p d^p \).

Let any breakdown \( x = yuvze \) with \( |uv| \leq p \), \( u, v, w, e \) be given. The compass must "touch" at least one region A, B, C, or D.

Out: • If it touches A, it can't also touch C. \( x^{(i)} = y u^i v w^i e \)
  • If it touches B, it can't also touch D. \( x^{(i)} = y u^i v w^i e \)
  • If it touches C, it can't also touch D. \( x^{(i)} = y u^i v w^i e \)

Hence in \( x^{(i)} \) or \( x^{(i)} \) or any \( x^{(i)} \) with \( i \neq i \), the restrictions \#u = \#c
  and \#b = \#d cannot both be present. So \( x^{(i)} \) \( \notin \) A, so A is not a CFL.

D Some proof works for \( A'' = \sum a^m b^n a^m b^n : m, n \geq 0 \).

D Then it also works for \( L_{WW} : w e i w b f^* i e d \) (for double words).
The argument is then easiest for \( i = 0 \). Another way: take \( R = d \) and use \( D \cap R = A'' \), so \( i \neq D \) were a CFL then \( A'' \) would be also.

D The "NR" idea helps somewhat with languages like \( L = a^i b^i c^i \)
  \( L'' = L \cap R = \sum a^m b^n c^p : \#a(x) > \#b(x) \) \& \#b(x) > \#c(x) \).

L'' = \( L'' \cap R = \sum a^m b^n c^p : m \geq n \) and \( n \geq p \). Enough to show \( L'' \) not a CFL

Take \( x = \) a\(^p\) b\(^p\) c\(^p+2\) and pump as \((i, k + e)\) if both \( e \), must pump as \((i, n + e)\) pump in b\(^k\).

If \( \lambda \) is only \( a \) or \( c, \) must pump down.
CFGs can handle nested and sequential binary dependencies:

\[ B_1 = \{ a^m b^n a^n b^m \mid m, n \geq 0 \} \]

is a CFL by nesting.

CFG

\[
\begin{align*}
S & \rightarrow aSbT \\
T & \rightarrow bTa \mid \epsilon
\end{align*}
\]

\[ B_2 = \{ a^m b^n a^n b^m : m, n \geq 0 \} \] is also a CFL, it is in fact equal to \( A \cdot A \) where \( A = \{ a^m b^n : m, n \geq 0 \} \).

What kind of machine can handle \( \{ a^n b^n \} \) and \( \{ a^n b^n c^n d^n \} \)?

What capabilities can we add to a DFA to handle these languages?

\[ x \sim \]

\[ \begin{array}{c}
\text{a}^n \text{?} \\
\text{?} \\
\text{b}^n \text{?} \\
\text{?} \\
\text{c}^n \text{?} \\
\text{?}
\end{array} \]

for some \( n \geq 0 \).

1. Allow the DFA to change a char on the tape.
2. Allow the DFA to move its read head \( L \) as well as \( R \).

(1) by itself makes no difference — the new char could be lumped in with a new state in \( \mathcal{Q} \times \mathcal{Z} = \mathcal{Q} \) CHANGED THM, NOT IN TEXT! Two-Way DFA crossing sequences.

(2) by itself also makes no difference. 

(1) \& (2) together create a Turing Machine: \( M = (\mathcal{Q}, \Gamma, \prod, \mathcal{S}, s, F) \)

where \( \mathcal{S} \subseteq (\mathcal{Q} \times \Gamma^*) \times (\prod \times Z \times Z \times Z \times Z \times Z \times Z \times Z \times Z \times Z \times Z) \).

\( (p, c \rightarrow d, D, q ) \)