Recall the **pumping lemma** for regular languages.

It told us that if there was a string long enough to cause a **cycle** in the DFA for the language, then we could **pump** the cycle and discover an **infinite sequence** of strings that had to be in the language.
Pumping Lemma for CFL’s: Intuition

- For CFL’s the situation is a little more complicated.
- We can always find two pieces of any sufficiently long string to pump in tandem.
- That is, if we repeat each of these two pieces the same number of times, we get another string in the language.
The CFL Pumping Lemma

Theorem
For every CFL $L$ there is an integer $n$, such that for every string $z$ in $L$ of length $\geq n$, there exists $z = uvwx$ such that:

- $|vwx| \leq n$.
- $|vx| > 0$.
- For all $i \geq 0$, $uv^iwx^i y \in L$. 

Proof of the Pumping Lemma

- Start with a CNF grammar for $L - \{\varepsilon\}$.
- Let the grammar have $m$ variables.
- Pick $n = 2^m$.
- Let $|z| \geq n$.
- We claim ("Lemma 1") that a parse tree with yield $z$ must have a path of length $m+2$ or more.
If all the paths in the parse tree of a CNF grammar are of length $\leq m + 1$, then the longest yield has length $2^{m-1}$, as in:
• Now we know that the parse tree for \( z \) has a path with at least \( m+1 \) variables.

• Consider some longest path.

• There are only \( m \) different variables, so among the lowest \( m+1 \) we can find two nodes with the same label, say \( A \).

• The parse tree thus looks like:
Proof of the Pumping Lemma
Non-CFL’s typically involve trying to match two pairs of counts or match two strings.

**Example:** Show that \( L = \{0^i10^i10^i \mid i \geq 1\} \) is not a CFL.

Proof using the pumping lemma.

Suppose \( L \) were a CFL.

Let \( n \) be \( L \)’s pumping length.
- Consider $z = 0^n10^n10^n$.
- We can write $z = uvwxy$, where $|vwx| \leq n$, and $|vx| \geq 1$.
- **Case 1**: $vx$ has no 0’s.
  - Then at least one of them is a 1, and $uwy$ has at most one 1, which no string in $L$ does.
Using the Pumping Lemma

- Still considering $z = 0^n10^n10^n$.
- **Case 2:** $vx$ has at least one 0.
  - $vwx$ is too short (length $\leq n$) to extend to all three blocks of 0’s in $0^n10^n10^n$.
  - Thus, $uwy$ has at least one block of $n$ 0’s, and at least one block with fewer than $n$ 0’s.
  - Thus, $uwy$ is not in $L$. 

Mridul Aanjaneya
Automata Theory
• As usual, when we talk about a CFL we really mean a representation for the CFL, e.g., a CFG or a PDA accepting by final state or empty stack.
• There are algorithms to decide if:
  • String $w$ is in CFL $L$.
  • CFL $L$ is empty.
  • CFL $L$ is infinite.
Many questions that can be decided for regular sets cannot be decided for CFL’s.

**Example:** Are two CFL’s the same?

**Example:** Are two CFL’s disjoint?

- How would you do that for regular languages?

Need theory of Turing Machines and decidability to prove no algorithm exists.
• We already did this.
• We learned to eliminate variables that generate no terminal string.
• If the start symbol is one of these, then the CFL is empty; otherwise not.
• Want to know if string $w$ is in $L(G)$.
• Assume $G$ is in CNF.
  • Or convert the given grammar to CNF.
  • $w = \varepsilon$ is a special case, solved by testing if the start symbol is nullable.
• Algorithm CYK is a good example of dynamic programming and runs in $O(n^3)$, where $n = |w|$.
• Let \( w = a_1a_2 \ldots a_n \).
• We construct an \( n \)-by-\( n \) triangular array of sets of variables.
• \( X_{ij} = \{ \text{variables } A \mid A \Rightarrow^* a_i \ldots a_j \} \).
• Induction on \( j-i+1 \).
  • The length of the derived string.
• Finally, ask if \( S \) is in \( X_{1n} \).
**CYK Algorithm**

- **Basis:** $X_{ii} = \{A \mid A \rightarrow a_i \text{ is a production}\}$.

- **Induction:** $X_{ij} = \{A \mid \text{there is a production } A \rightarrow BC \text{ and an integer } k, \text{ with } i \leq k < j, \text{ such that } B \text{ is in } X_{ik} \text{ and } C \text{ is in } X_{k+1,j}\}$. 
Example: CYK Algorithm

- Grammar: $S \rightarrow AB$, $A \rightarrow BC|a$, $B \rightarrow AC|b$, $C \rightarrow a|b$
- String $w = ababa$.
- $X_{11} = \{A,C\}$, $X_{22} = \{B,C\}$, $X_{33} = \{A,C\}$, $X_{44} = \{B,C\}$, $X_{55} = \{A,C\}$.
- $X_{12} = \{B,S\}$, $X_{23} = \{A\}$, $X_{34} = \{B,S\}$, $X_{45} = \{A\}$.
- $X_{13} = \{A\}$, $X_{24} = \{B,S\}$, $X_{35} = \{A\}$.
- $X_{14} = \{B,S\}$, $X_{25} = \{A\}$.
- $X_{15} = \{A\}$. 
Testing Infiniteness

- The idea is essentially the same as for regular languages.
- Use the pumping length $n$.
- If there is a string in the language of length between $n$ and $2n-1$, then the language is infinite; otherwise not.
Closure Properties of CFL’s

- CFL’s are **closed** under union, concatenation, and Kleene closure.
- Also, under reversal, homomorphisms and **inverse homomorphisms**.
- But **not** under intersection or difference.
Closure of CFL’s under Union

- Let $L$ and $M$ be CFL’s with grammars $G$ and $H$, respectively.
- Assume $G$ and $H$ have no variables in common.
  - Names of variables do not affect the language.
- Let $S_1$ and $S_2$ be the start symbols of $G$ and $H$. 
Closure of CFL’s under Union

- Form a new grammar for $L \cup M$ by combining all the symbols and productions of $G$ and $H$.
- Then, add a new start symbol $S$.
- Add the production $S \rightarrow S_1 | S_2$. 
• In the new grammar, all derivations start with $S$.
• The first step replaces $S$ by either $S_1$ or $S_2$.
• In the first case, the result must be a string in $L(G) = L$, and in the second case a string in $L(H) = M$. 
Closure of CFL’s under Concatenation

- Let $L$ and $M$ be CFL’s with grammars $G$ and $H$, respectively.
- Assume $G$ and $H$ have no variables in common.
- Let $S_1$ and $S_2$ be the start symbols of $G$ and $H$. 
• Form a new grammar for LM by combining all the symbols and productions of G and H.
• Add a new start symbol S.
• Add the production $S \rightarrow S_1S_2$.
• Every derivation from S results in a string in L followed by one in M.
Closure under Star

- Let $L$ have grammar $G$, with start symbol $S_1$.
- Form a new grammar for $L^*$ by introducing to $G$ a new start symbol $S$ and the productions $S \rightarrow S_1S \mid \varepsilon$.
- A rightmost derivation from $S$ generates a sequence of zero or more $S_1$’s, each of which generates some string in $L$. 

Mridul Aanjaneya
If \( L \) is a CFL with a grammar \( G \), form a grammar for \( L^R \) by reversing the right side of every production.

**Example:** Let \( G \) have \( S \rightarrow 0S1 \mid 01 \).

The reversal of \( L(G) \) has grammar \( S \rightarrow 1S0 \mid 10 \).
Closure of CFL’s under Homomorphisms

- Let $L$ be a CFL with a grammar $G$.
- Let $h$ be a homomorphism on the terminal symbols of $G$.
- Construct a grammar for $h(L)$ by replacing each terminal symbol $a$ by $h(a)$. 
Example: Closure under Homomorphisms

- $G$ has productions $S \rightarrow 0S1 \mid 01$.
- $h$ is defined by $h(0) = ab$, $h(1) = \varepsilon$.
- $h(L(G))$ has the grammar with productions $S \rightarrow abS \mid ab$. 
Closure under Inverse Homomorphisms

- Here, grammars do not help us.
- But a PDA construction serves nicely.
- **Intuition:** Let $L = L(P)$ for some PDA $P$.
- Construct PDA $P'$ to accept $h^{-1}(L)$.
- $P'$ simulates $P$, but keeps, as one component of a two-component state a buffer that holds the result of applying $h$ to one input symbol.
Architecture of $P'$

- Read **first remaining symbol** in buffer as if it were **input** to $P$.

Input: 0 0 1 1

<h(0)> 

Buffers

State of $P$

Stack of $P$
States are pairs $[q, b]$, where:

1. $q$ is a state of $P$.
2. $b$ is a suffix of $h(a)$ for some symbol $a$.

Thus, only a finite number of possible values for $b$.

Stack symbols of $P'$ are those of $P$.

Start state of $P'$ is $[q_0, \varepsilon]$. 
Formal Construction of $P'$

- Input symbols of $P'$ are the symbols to which $h$ applies.
- Final states of $P'$ are the states $[q, \varepsilon]$ such that $q$ is a final state of $P$. 
Transitions of $P'$

1. $\delta'([q, \varepsilon], a, X) = \{([q, h(a)], X)\}$ for any input symbol $a$ of $P'$ and any stack symbol $X$.
   - When the buffer is empty, $P'$ can reload it.

2. $\delta'([q, bw], \varepsilon, X)$ contains $([p, w], \alpha)$ if $\delta(q, b, X)$ contains $(p, \alpha)$, where $b$ is either an input symbol of $P$ or $\varepsilon$.
   - Simulate $P$ from the buffer.
Proving Correctness of $P'$

- We need to show that $L(P') = h^{-1}(L(P))$.
- **Key argument:** $P'$ makes the transition $([q, \varepsilon], w, Z_0) \vdash^* ([q, x], \varepsilon, \alpha)$ if and only if $P$ makes transition $(q_0, y, Z_0) \vdash^* (q, \varepsilon, \alpha)$, $h(w) = yx$, and $x$ is a suffix of the last symbol of $w$.
- Proof in both directions is an induction on the number of moves made.
  - Left as exercises.
Unlike the regular languages, the class of CFL’s is not closed under intersection.

We know that \( L_1 = \{0^n1^n2^n | n \geq 1\} \) is not a CFL (using the pumping lemma).

However, \( L_2 = \{0^n1^n2^i | n \geq 1, i \geq 1\} \) is.

- CFG: \( S \rightarrow AB, \ A \rightarrow 0A1|01, \ B \rightarrow 2B|2 \).

So is \( L_3 = \{0^i1^n2^n | n \geq 1, i \geq 1\} \).

But \( L_1 = L_2 \cap L_3 \).
We can prove something more general:

- Any class of languages that is closed under difference is closed under intersection.

**Proof:** $L \cap M = L - (L - M)$.

Thus, if CFL’s were closed under difference, they would be closed under intersection, but they are not.
• Intersection of two CFL’s need not be context-free.
• But the intersection of a CFL with a regular language is always a CFL.
• Proof involves running a DFA in parallel with a PDA, and noting that the combination is a PDA.
  • PDA’s accept by final state.
• Let the DFA $A$ have transition function $\delta_A$.
• Let the PDA $P$ have transition function $\delta_P$.
• States of combined PDA are $[q,p]$, where $q$ is a state of $A$ and $p$ is a state of $P$.
• $\delta([q,p],a,X)$ contains $([\delta_A(q,a),r],\alpha)$ if $\delta_P(p,a,X)$ contains $(r,\alpha)$.
  - **Note:** $a$ could be $\varepsilon$, in which case $\delta_A(q,a) = q$. 
• Accepting states of combined PDA are those \([q,p]\) such that \(q\) is an accepting state of \(A\) and \(p\) is an accepting state of \(P\).

• **Easy induction:** \(([q_0,p_0],w,Z_0) \vdash^* ([q,p],\varepsilon,\alpha)\) if and only if \(\delta_A(q_0,w) = q\) and in \(P\): \((p_0,w,Z_0) \vdash^* (p,\varepsilon,\alpha)\).