Kleene’s Theorem: Let $A \subseteq \Sigma^*$ be any language. Then the following are equivalent:

1. $A = \mathcal{L}(D)$, for some DFA $D$.
2. $A = \mathcal{L}(N)$, for some NFA $N$ wo $\epsilon$ transitions
3. $A = \mathcal{L}(N)$, for some NFA $N$.
4. $A = \mathcal{L}(e)$, for some regular expression $e$.
5. $A$ is regular.

Myhill-Nerode Theorem: The language $A$ is regular iff $\sim_A$ has a finite number of equivalence classes. Furthermore, this number of equivalence classes is equal to the number of states in the minimum-state DFA that accepts $A$. 
Closure Theorem for Regular Sets: Let \( A, B \subseteq \Sigma^* \) be regular languages and let \( h : \Sigma^* \to \Gamma^* \) and \( g : \Gamma^* \to \Sigma^* \) be homomorphisms. Then the following languages are regular:

1. \( A \cup B \)
2. \( AB \)
3. \( \overline{A} = (\Sigma^* - A) \)
4. \( A \cap B \)
5. \( h(A) \)
6. \( g^{-1}(A) \)
Proofs of Closure Properties

Because we have so many equivalent models for the class of regular languages, we can pick the one that makes each proof easiest:

1. Regular Expressions: union, concatenation, star
2. DFA: complement, hence intersection
3. (product of DFA’s gives intersection directly)
4. forward homomorphism: substitute into regexp or general NFA
5. inverse homomorphism: simulate on DFA
6. reversal: see HW#1
Let \( h : \Sigma^* \rightarrow \Delta^* \) be a homomorphism.

If \( R \) is a regular expression over \( \Sigma \), we can compute \( h(R) \) inductively. Then \( h(\mathcal{L}(R)) = \mathcal{L}(h(R)) \).

If \( M \) is a DFA with alphabet \( \Delta \) and \( \mathcal{L}(M) = A \), we can make a DFA for \( h^{-1}(A) \) as follows. States, start, and final states are the same as \( M \). For every letter \( a \) in \( \Sigma \) and every state \( q \), define \( \delta(q, a) \) to be \( \delta^*_M(q, h(a)) \). Then for any \( w \in \Sigma^* \), \( \delta^*(q, w) = \delta^*_M(q, h(w)) \), and \( \delta^*(q_0, w) \in F \) iff \( \delta^*_M(q_0, h(w)) \in F \) iff \( h(w) \in A \) iff \( w \in h^{-1}(A) \).
Definition: A context-free grammar (CFG) is a 4-tuple $G = (V, \Sigma, R, S)$,

- $V = \text{variables} = \text{nonterminals}$,
- $\Sigma = \text{terminals}$,
- $R = \text{rules} = \text{productions}$, $R \subseteq V \times (V \cup \Sigma)^*$,
- $S \in V$,
- $V, \Sigma, R$ are all finite.
\[ G_1 = (\{S\}, \{a, b\}, R_1, S) \]
\[ R_1 = \{(S, aSb), (S, \epsilon)\} = \{S \to aSb|\epsilon\} \]

\[
\begin{align*}
S & \to \epsilon \\
S & \to aSb \Rightarrow ab \\
S & \to aSb \Rightarrow aaSbb \Rightarrow aabb \\
S & \to aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb \\
\end{align*}
\]

\[
\mathcal{L}(G_1) = \{w \in \{a, b\}^* \mid S \xrightarrow{G_1} w\} \\
= \{a^n b^n \mid n \in \mathbb{N}\} \\
\mathcal{L}(G) = \{w \in \Sigma^* \mid S \xrightarrow{G} w\}
\]
\[ G_2 = \]
\[(\{ E, T, F, V, L, D, C \}, \{(\ ), +, *, x, y, z, 0, 1, \ldots, 9 \}, R_2, E)\]

\[ R_2 = \]
\[
E \rightarrow E + T | T \\
T \rightarrow T \star F | F \\
F \rightarrow (E) | V | C \\
V \rightarrow LD
\]

\textbf{Parse Tree:}
Pumping Lemma for Regular Sets: Let $D = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Let $n = |Q|$. Let $w \in \mathcal{L}(D)$ s.t. $|w| \geq n$. Then $\exists x, y, z \in \Sigma^*$ s.t. the following all hold:

- $xyz = w$
- $|xy| \leq n$
- $|y| > 0$, and
- $(\forall k \geq 0)xy^kz \in \mathcal{L}(D)$

Proof: Let $w \in \mathcal{L}(D)$ s.t. $|w| \geq n$.

\[
w = w_1 \cdots w_n u \quad q_0 \quad q_1 \quad q_2 \quad q_3 \quad \cdots \quad q_{n-1} \quad q_n
\]

By the Pigeonhole Principle, $(\exists i < j)q_i = q_j$

\[
w = \underbrace{w_1 \cdots w_i}_{x} \underbrace{w_{i+1} \cdots w_j}_{y} \underbrace{w_{j+1} \cdots w_n u}_{z} \quad q_0 \quad q_i \quad q_{i} \quad q_{i} \quad q_f
\]

$\delta^*(q_i, y) = q_i$. Thus, $xy^kz \in \mathcal{L}(D)$ for all $k \in \mathbb{N}$. ♠️
Prop: \( E = \{ a^r b^r \mid r \in \mathbb{N} \} \) is not regular.

Proof: Suppose that \( E \) were regular, accepted by a DFA with \( n \) states. Let \( w = a^n b^n \).

By pumping lemma, \( w = a^n b^n = xyz \) where

- \( |xy| \leq n \)
- \( |y| > 0 \), and
- \( (\forall k \in \mathbb{N}) xy^k z \in E \)

Since \( 0 < |xy| \leq n \), \( y = a^i, 0 < i \leq n \).

Thus \( xy^0 z = a^{n-i} b^n \in E \).

But, \( a^{n-i} b^n \not\in E \).

\( \Rightarrow \Leftarrow \)

Therefore \( E \) is not regular.  

\( \blacksquare \)
CFL Pumping Lemma: Let $A$ be a CFL. Then there is a constant $n$, depending only on $A$ such that if $z \in A$ and $|z| \geq n$, then there exist strings $u, v, w, x, y$ such that,

- $z = uwx$, and
- $|vx| \geq 1$, and
- $|wx| \leq n$, and
- for all $k \in \mathbb{N}, uv^kwx^ky \in A$
proof:  Let $G = (V, \Sigma, R, S)$ be a CFG with $\mathcal{L}(G) = A$. Let $n$ be so large that for $|z| \geq n$ s.t. $N \stackrel{*}{\Rightarrow}_{G} z$ for some $N \in V$, the parse tree for $z$ has height $> |V| + 2$.

Let $z \in A$, $|z| \geq n$.

The parse tree for $z$ has height greater than $|V| + 2$.

Thus, some path repeats a nonterminal, $N$.

$z = uvwxy$; $(\forall k \in \mathbb{N})(uv^kwx^ky \in A)$
**Prop:** \( P = \{a^n b^m a^n b^m \mid n, m \in \mathbb{N}\} \) is not a CFL.

**Proof:** Suppose \( P \) were a CFL.

Let \( n \) be constant of the pumping lemma.

Let \( z = a^n b^n a^n b^n \).

By pumping lemma, \( z = uvwxy \), and

1. \(|vx| \geq 1|,

2. \(|vwx| \leq n|, and

3. for all \( k \in \mathbb{N}, uv^kwx^k y \in P \)

Since \(|vwx| \leq n|, \quad vw x \in a^*b^* \) or \( vw x \in b^*a^* \).

If either \( v \) or \( x \) contains both \( a \)'s, and \( b \)'s, then \( uv^2 wx^2 y \) is not in \( P \).

Suppose that \( vx \) contains at least one \( a \). Then, \( uv^2 wx^2 y \) is not in \( P \), because it has more \( a \)'s in one group than the other.

Suppose that \( vx \) contains at least one \( b \). Then, \( uv^2 wx^2 y \) is not in \( P \), because it has more \( b \)'s in one group than the other.

Thus, \( uv^2 wx^2 y \) is not in \( P \).

\( \Rightarrow \Leftrightarrow \quad \text{Thus P is not a CFL.} \)
Prop: \( \text{NONCFL} = \{a^n b^n c^n : n \in \mathbb{N}\} \) is not a CFL.

Proof:

The argument is almost identical. We let \( z = a^n b^n c^n \) where \( n \) is larger than the constant given by the CFL Pumping Lemma. So \( z = uvwxy \) with \( |vwx| > 0 \), \( |vwx| \leq n \), and \( uv^iwx^i y \) in \( \text{NONCFL} \) for all \( i \). Again, neither \( v \) nor \( x \) can contain letters of two different types, or \( uv^2wx^2y \) is not in \( a^*b^*c^* \). But then \( uv^2wx^2y \) cannot contain equal numbers of \( a \)'s, \( b \)'s, and \( c \)'s, as only one or two types of letter have been added.

\( \blacklozenge \)

There are languages that are not CFL’s but that still satisfy the conclusion of the CFL Pumping Lemma. On HW #1 you will prove one of these to be a non-CFL, using closure properties.
Definition: A pushdown automaton (PDA) is a 7-tuple, \( P = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F) \)

- \( Q \) = finite set of states,
- \( \Sigma \) = input alphabet,
- \( \Gamma \) = stack alphabet,
- \( \Delta \subseteq (Q \times \Sigma^* \times \Gamma^*) \times (Q \times \Gamma^*) \) finite set of transitions,
- \( q_0 \in Q \) start state,
- \( Z_0 \in \Gamma \) initial stack symbol,
- \( F \subseteq Q \) final states.

\[
PDA = NFA + \text{stack}
\]

\[
\mathcal{L}(P) = \{ w \in \Sigma^* \mid (q_0, Z_0) \xrightarrow{w}_P (q, X), q \in F, X \in \Gamma^* \}
\]
\[ P_1 = (\{q, r, s\}, \{a, b\}, \{A, B, Z_0\}, \Delta_1, q, Z_0, \{s\}) \]

\[ \Delta_1 = \{(q, a, \epsilon), (q, A)\}, \{(q, b, \epsilon), (q, B)\}, \{(q, \epsilon, \epsilon), (r, \epsilon)\}, \{(r, a, A), (r, \epsilon)\}, \{(r, b, B), (r, \epsilon)\}, \{(r, \epsilon, Z_0), (s, \epsilon)\} \}

\[ \mathcal{L}(P_1) = \{ww^R \mid w \in \{a, b\}^*\} \]
**Theorem 3.1** Let $A \subseteq \Sigma^*$ be any language. Then the following are equivalent:

1. $A = \mathcal{L}(G)$, for some CFG $G$.
2. $A = \mathcal{L}(P)$, for some PDA $P$.
3. $A$ is a context-free language.

**Proof:** See [HMU] or [LP] or [S]. To prove (1) implies (2), we build a “bottom-up parser” or “top-down” parser, similar to those used in real-world compilers except that the latter are deterministic. The proof that (2) implies (1) is of less practical interest – it defines languages like “the strings that could take state $i$ and empty stack to state $j$ and empty stack” and shows that these languages are interrelated by CFG rules.