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 $N$ without $\epsilon$ 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 $h : \Sigma^* \rightarrow \Gamma^*$ and $g : \Gamma^* \rightarrow \Sigma^*$ 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. $g^{-1}(A)$

A homomorphism of strings is a function $g$ such that for any strings $u$ and $v$, $g(uv) = g(u)g(v)$. The set $g^{-1}(A)$ is defined as $\{u : g(u) \in A\}$. 
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: easy by regular expressions, but also doable with NFA’s (exercise)
Let $h : \Sigma^* \rightarrow \Delta^*$ be a homomorphism.

If $R$ is a regular expression over $\Sigma$, we can compute a regular expression $h(R)$ by induction on the definition of regular expressions. Then we can prove by induction that $h(\mathcal{L}(R)) = \mathcal{L}(h(R))$.

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.
\[ G_1 = (\{S\}, \{a, b\}, R_1, S) \]
\[ R_1 = \{\langle S, aSb\rangle, \langle S, \epsilon\rangle\} = \{S \rightarrow aSb|\epsilon\} \]

\[ S \rightarrow \epsilon \]
\[ S \rightarrow aSb \Rightarrow ab \]
\[ S \rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb \]
\[ S \rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aabbbb \]

\[ \mathcal{L}(G_1) = \{w \in \{a, b\}^* \mid S \xrightarrow{G_1} w\} \]
\[ = \{a^n b^n \mid n \in \mathbb{N}\} \]

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

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

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. $|w| \geq n$.

$$w = w_1 \ldots w_n \ldots$$

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

$$w = w_1 \ldots w_i \underbrace{w_{i+1} \ldots w_j}_{q_i} \underbrace{w_{j+1} \ldots w_n}_{q_{i+1}}$$

$\delta^*(q_i, y) = q_i$. Thus, $xy^kz \in \mathcal{L}(D)$ for all $k \in \mathbb{N}$. ♠

8
We showed: $E = \{a^r b^r \mid r \in \mathbb{N}\}$ is not regular.

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

By the pumping lemma, $w = a^n b^n = xyz$ where

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

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

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

But, $a^{n-i}b^n \notin E$.

$\Rightarrow \Leftarrow$

Therefore $E$ is not regular.
**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:

- $z = uvwxy$, and
- $|vx| \geq 1$, and
- $|vwx| \leq n$, 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, \( N \).

\[
z = uvwx; \quad (\forall k \in \mathbb{N})(uv^kwx^ky \in A)
\]
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 the constant of the CFL pumping lemma.

Let $z = a^n b^n a^n b^n$.

By the CFL pumping lemma, $z = uvwx y$, and

1. $|vx| \geq 1$,
2. $|vwx| \leq n$, and
3. for all $k \in \mathbb{N}$, $uv^k wx^k y \in P$

Since $|vwx| \leq n$, $vwx \in a^* b^*$ or $vwx \in b^* a^*$.

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 $P$.

$\implies \Leftarrow$ Thus $P$ is not a CFL.
Prop: \( NONCFL = \{a^n b^n c^n : n \in \mathbb{N}\} \) 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 \( uv^iwx^iy \) in \( NONCFL \) 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.

\( \blacklozenge \)
Any CFL satisfies the conclusion of the CFL Pumping Lemma, but it is not true that any non-CFL must fail to satisfy it. There are other tools that can show a language to be a non-CFL. These include stronger forms of the Pumping Lemma and more closure properties.

Let $EQUAL$ be the set of strings in $(a \cup b \cup c)^*$ that have an equal number of $a$’s, $b$’s, and $c$’s. You can use the CFL Pumping Lemma on this with the right choice of $z$, but far easier is using the fact that the intersection of $EQUAL$ with $a^*b^*c^*$ is the language $NONCFL$.

If $A$ is a CFL and $R$ a regular language, then $A \cup R$ must be regular. Proving this, however, requires a different characterization of the CFL’s.
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.

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

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

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

$L(P_1) = \{ww^R \mid w \in \{a, b\}^*\}$
Theorem 4.1 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: We give only a sketch here – there are detailed proofs in [HMU], [LP], and [S].

To prove (1) implies (2), we can build a “bottom-up parser” or “top-down” parser, similar to those used in real-world compilers except that the latter are deterministic.
The top-down parser is a PDA that:

- begins by pushing “$S$” onto its stack
- may pop a terminal from the stack if can at the same time read a matching input letter,
- may execute a rule $A \rightarrow w$ by popping $A$ and pushing $w^R$,
- ends by popping the $\$\$ when done with the input

The bottom-up parser, somewhat similarly:

- pushes “$\$” onto its stack,
- may transfer a terminal from the input to the stack,
- may execute $A \rightarrow w$ by popping $w$ and pushing $A$,
- ends by popping $S\$\$ when done
The proof that the language of any PDA is a CFL (that (2) implies (1)) is of less practical interest.

Given states $i$ and $j$, let $A_{ij}$ be the set of strings that could take the PDA from state $i$ with empty stack to state $j$ with empty stack.

If we can define rules making each $A_{ij}$ a CFL we win, because the language of the PDA is the union of $A_{sf}$ for all final states $f$, where $s$ is the start state. (So our grammar has a rule $S \rightarrow A_{sf}$ for each $f$.)

We have all rules of the form $A_{pq} \rightarrow A_{pr}A_{rq}$, and a rule $A_{pq} \rightarrow aA_{rs}b$ whenever moves of the PDA warrant it.

Here I am skipping some assumptions on the PDA, and the (nontrivial) proof that any accepting run of the PDA corresponds to a valid derivation in our grammar.