The context-free languages are used far less often than the regular languages or the most general computational level. Nonetheless, they are worth at least a quick look. These languages cover those like \( 0^n1^n1^n2^n0^n3^n \). Note that all regular languages will be context-free.

There are two ways we describe context-free languages. The first is with context-free grammars, an analog to regular expressions. Compilers and parsers typically have a context-free grammar for their language, although some are context-sensitive or worse.

**Ex)**

\[
\begin{align*}
A & \rightarrow 0A1 \\
A & \rightarrow \varepsilon \\
B & \rightarrow AA
\end{align*}
\]

Starting with just \( A \), we get

\[
A \rightarrow \varepsilon \text{ or } 0A1 \rightarrow 01 \text{ or } 00A11 \rightarrow \ldots
\]

Starting with \( B \), we get \( 0^n1^n0^n1^n \).

A grammar consists of substitution rules (also known as productions), variables like \( A \), and terminals like \( 0 \) or \( 1 \). The sequence \( A \rightarrow 0A1 \rightarrow 00A11 \rightarrow 001111 \rightarrow 001111 \) is called a derivation. We can also represent this as a parse tree.

The set of all strings a grammar produces is the language of the grammar. A language which a (context-free) grammar produces is context-free. We can combine rules with an or to get \( A \rightarrow 0A1 | \varepsilon \).
Sipser gives us a good example.

\[
\begin{align*}
S & \rightarrow NP \ VP \\
NP & \rightarrow CN \mid CN \ PP \\
VP & \rightarrow CV \mid CV \ PP \\
PP & \rightarrow P \ CN \\
CN & \rightarrow A \ N \\
CV & \rightarrow V \mid V \ NP \\
A & \rightarrow a \mid the \\
N & \rightarrow boy \mid girl \mid flower \\
V & \rightarrow touches \mid likes \mid sees \\
P & \rightarrow with
\end{align*}
\]

\[
S \rightarrow NP \ VP \\
\rightarrow CN \ CV \\
\rightarrow A \ N \ V \ NP \\
\rightarrow \text{the boy sees} \ CN \ PP \\
\rightarrow \text{the boy sees} \ A \ N \ P \ CN \\
\rightarrow \text{the boy sees} \ a \ girl \ with \ A \ N \\
\rightarrow \text{the boy sees} \ a \ girl \ with \ a \ flower
\]

Note that English is not a context-free language. Most such languages are not, though you can get far with just context-free grammars.
A context-free language can also be represented with a pushdown automata.

A PDA is a finite state machine with a stack. The stack can push and pop items onto the top of the stack.

Ex) If our alphabet is \( \mathbb{Z}, 0, 1, \epsilon, +, -, \times, \div, (, ) \), then we can validate arithmetic expressions. We push (and ops onto the stack and pop off when we get a) or a second number respectively. We accept if the stack is clean when we're done.

Note that PDAs have no mechanism for checking if the stack is empty. However, you can push a unique symbol onto it to denote the bottom of the stack.

Ex) We draw a PDA for \( \{0^n 1^n | n \geq 0\} \).

Note that this is a nondeterministic PDA. Unlike with regular languages, where DFAs and NFAs are equivalent, NPDA's are more powerful than DPDAs.
Closure Properties

- Union
- Reversal
- Concatenation
- Kleene Star
- Not: intersection or complement

Exj \( A = \{ a^n b^n c^m \mid n, m \in \mathbb{Z} \} \) \( B = \{ a^n b^n c^n \mid n \in \mathbb{N} \} \)

\[
\begin{align*}
S &\rightarrow LR \\
L &\rightarrow aLb \\
L &\rightarrow \varepsilon \\
R &\rightarrow cR \text{ (or } Rc) \\
R &\rightarrow \varepsilon \\
S &\rightarrow LR \\
L &\rightarrow aL \text{ (or } La) \\
L &\rightarrow \varepsilon \\
R &\rightarrow bRc \\
R &\rightarrow \varepsilon \\
\end{align*}
\]

Exj \( A \cap B = \{ a^n b^n c^n \mid n \geq 0 \} \) is not context-free.

Context-Free Pumping Lemma

Let \( L \subseteq \Sigma^* \) be context-free. Then \( \exists p \geq 1 \) such that \( \forall w \in L \)
of length at least \( p \), we can write \( w = \alpha \gamma v \) such that

i) \( |\gamma| \geq 1 \)

ii) \( |\alpha \gamma| \leq p \)

iii) \( \alpha \gamma v \gamma v^n \in L \) for all \( n \geq 0 \).

In \( A \cap B \) above, the trouble lies in requiring all of \( a^n b^n \) and \( c^n \) to have the same length. The pumping lemma only lets us match two of them at a time via \( u \) and \( v \).
Example: $\exists w w^R | w \in \{0, 1\}^*$

- $S \rightarrow 0S0$
- $S \rightarrow 1S1$
- $S \rightarrow \varepsilon$

Diagram:

- States: $q_0, q_1, q_2$
- Transitions:
  - $q_0$:
    - $\varepsilon, \varepsilon \rightarrow 0$
    - $0, 0 \rightarrow 0$
    - $0, 1 \rightarrow \varepsilon$
  - $q_1$:
    - $\varepsilon, \varepsilon \rightarrow 0$
    - $0, 0 \rightarrow \varepsilon$
    - $1, 1 \rightarrow \varepsilon$
  - $q_2$:
    - $\varepsilon, \varepsilon \rightarrow 0$
    - $0, 0 \rightarrow \varepsilon$
    - $1, 1 \rightarrow \varepsilon$

Word: $w = 010010$

Some solution but $q_0$ is no longer an accepting state.