The Chomsky Hierarchy

There are other types of grammars.
We noted that every regular language has a CFG. In fact, a regular language (without $\varepsilon$) can be generated by a special form:

A *regular grammar* is one where every production is of the form $A \rightarrow bC$ or $A \rightarrow a$ (where $a$ and $b$ are some terminals and $C$ some variable).
Theorem. Every regular language is generated by a regular grammar.

Proof. Idea: produce a grammar such that a derivation mimics the operation of the automaton.

Every stage of derivation will have a single variable that is the state of the FA.
Construction of Regular Grammar

Start with DFA for the language. Introduce one variable for each state.

For each transition, add a production: if $\delta(A, x) = B$, then add production $A \rightarrow xB$.

For each transition ending at an accept state, add a further production: if $B$ is accept state in above, then also add production $A \rightarrow x$.

The start variable is the start state.
Consider RE \((11 + 00)^* 11\). Here is NFA:

![NFA Diagram]

This yields the following regular grammar:

\[
\begin{align*}
S &\rightarrow 0C \mid 1A \\
A &\rightarrow 1B \mid 1 \\
B &\rightarrow 0C \mid 1A \\
C &\rightarrow 0S
\end{align*}
\]
Draw an FA, and from there write down a regular grammar, for the language given by the RE $00^*11^*$. 
Solution to Practice

\[ S \rightarrow 0A \]
\[ A \rightarrow 0A \mid 1B \mid 1 \]
\[ B \rightarrow 1B \mid 1 \]
In **unrestricted grammars**, productions have form \( u \rightarrow v \) where \( u \) and \( v \) are any strings of terminals and/or variables.

In **context-sensitive grammars**, productions have form \( xA \rightarrow xyz \) where \( x, y \) and \( z \) are strings of terminals and/or variables, and \( A \) is a variable.
A context-sensitive grammar for \(0^n 1^n 2^n\) is not obvious!

\[
S \rightarrow 0BS2 \mid 012 \\
B0 \rightarrow 0B \\
B1 \rightarrow 11
\]

(Try to derive \(000111222\).)
Chomsky introduced the hierarchy of grammars in his study of natural languages.

0. Unrestricted grammars.
2. Context-free grammars.
3. Regular grammars.

We have seen that regular grammars are accepted by FAs, and that CFGs are accepted by PDAs. We will see later machines for the other two types.
A variable is **usable** if it generates some string of terminals. A variable is **nullable** if it generates the empty string.

Example: In the following, $A$ and $B$ are usable but only $B$ is nullable.

\[
A \rightarrow 0A \mid 1B \mid 2C \\
B \rightarrow 0B \mid \varepsilon \\
C \rightarrow 1C
\]
**Algorithm for Nullable Variables**

**Identification of nullable variables.** Initialize all variables as not-nullable. 
Repeat:
  - go through all productions, and if any has RHS empty or all entries nullable, 
  - then mark the LHS variable as nullable; 
Until no increase in the set of nullable variables.

A similar procedure can be used to determine the usable variables.
A regular grammar is one where every production has the form $A \rightarrow bC$ or $A \rightarrow a$. The Chomsky hierarchy also includes context-sensitive grammars and unrestricted grammars.