Grammars

To describe a grammar we use two alphabets and a collection of rules known as productions.

(1) $T$ is the alphabet of terminal symbols.

(2) $N$ is the alphabet of non-terminal symbols, and we assume that $T$ and $N$ have no symbols in common.

(3) $S \in N$ is the start symbol.

(4) $P$ is the set of productions; they have the form $\alpha \rightarrow \beta$ where $\alpha$ and $\beta$ are strings constructed from the terminal and nonterminal symbols.

(5) The string $\alpha$ must contain at least one nonterminal symbol.

Thus a grammar can be thought of as a 4-tuple

$$G = (N, T, S, P).$$
The language of a grammar

From the grammar $G$ we can produce strings in the terminal alphabet $T$.

The language, $L(G)$, produced by the grammar $G$ is the collection of strings that can be produced from the start symbol by a sequence of productions.

The application of a sequence of productions is called a derivation.

The alphabet $T$ of terminal symbols of the grammar is the alphabet of the language $L(G)$. We produce the strings in $L(G)$ as follows.

- Begin by writing down $S$.

- At any stage replace a symbol which occurs to the left of the arrow of a production by the string to the right of that arrow.

- Continue until only terminal symbols remain in the string.
Example. Consider the grammar $\mathcal{G} = (\mathcal{N}, \mathcal{T}, S, \mathcal{P})$ where $\mathcal{N} = \{S, A, B\}$, $\mathcal{T} = \{0, 1\}$ and the productions are

$$
S \rightarrow ASB \\
S \rightarrow AB \\
A \rightarrow 0 \\
B \rightarrow 1.
$$

The language $L(\mathcal{G})$ of $\mathcal{G}$ is the set of strings

$$
\underbrace{00 \cdots 0}_{m} \underbrace{11 \cdots 1}_{m} \quad m \geq 1
$$

That is, $L(\mathcal{G}) = \{01, 0011, 000111, \ldots \}$.

This example shows that the language produced by a grammar **need not** be regular.

Productions may use the empty string $\varepsilon$ and so the grammar above can be written more simply as

$$
S \rightarrow 0A1 \\
A \rightarrow 0A1 \\
A \rightarrow \varepsilon.
$$

Note that the empty string is not in the language.
A grammar for a language

In general it is a difficult problem to find a grammar that produces a given language. In some cases, such as natural languages like English, Chinese, French or German, it is not even clear what constitutes the language. On the other hand, computer languages are generally easier to deal with and grammars are routinely used to construct compilers and interpreters for such languages.

The best we can do at this stage is to consider a simple example.
Example.

Let $L$ be the language

$$L = \{aaa\} \{aaa\}^*.$$

The language consists of all strings of a multiple of 3 $a$'s: $aaa$, $aaaaaa$, $aaaaaaa$, .... So our aim is to produce the strings $aaa$, $aaaaaa$, $aaaaaaa$, .... We can use the start symbol $S$ to produce the string $aaa$ ($S \rightarrow aaa$) and then use the production $S \rightarrow aaaS$ to produce any number of $aaa$'s. Hence a grammar for $L$ is $G = (\mathcal{N}, \mathcal{T}, S, \mathcal{P})$, where $\mathcal{N} = \{S\}$, $\mathcal{T} = \{a\}$ and the productions are

$$S \rightarrow aaa$$

$$S \rightarrow aaaS.$$

The same language can be produced by many different grammars.
Grammars and Regular Languages

It turns out that any regular language can be produced by a grammar with the following simple structure. The only productions are of the form

\[ \alpha \to a\beta \text{ or } \alpha \to a, \]

where \(\alpha\) and \(\beta\) are single non-terminal symbols and \(a\) is either a terminal symbol or \(\varepsilon\). Conversely, the language produced by such a grammar is regular.

The grammars in which all the productions are of the form \(\alpha \to \beta\), where \(\alpha\) is a single non-terminal symbol, are called context-free grammars.
Let $L$ be the language designated by the regular expression

$$(0 + 10)^*1,$$

Then $L$ consists of those strings which end with a 1 and in which every other 1 is followed by a 0. Let $\ell_n$ be the number of strings in $L$ of length $n$. If a string of $L$ begins with 0 it must have the form $0u$, where $u \in L$. If it begins with 1, it is either 1 itself or it has the form $10v$, where $v \in L$. Thus, for $n \geq 2$ we find that

$$\ell_n = \ell_{n-1} + \ell_{n-2}.$$

This is the recurrence relation which is satisfied by the Fibonacci numbers $F_n$ and we have the initial conditions $\ell_1 = 1$, $\ell_2 = 1$. Therefore $\ell_n = F_{n-1}$. 

\textbf{Fibonacci numbers}
We can see that $L$ is exactly the language accepted by the DFA:

The description of the strings of $L$ just given shows that $L$ is the language produced by the grammar $G = (\mathcal{N}, \mathcal{T}, S, \mathcal{P})$, where $\mathcal{N} = \{S\}$, $\mathcal{T} = \{0, 1\}$ and the productions are

\[
\begin{align*}
S & \rightarrow 0S \\
S & \rightarrow 10S \\
S & \rightarrow 1.
\end{align*}
\]