2.0 Formal Grammars

<u>Context-Free Grammar</u> (CFG). A language generator consisting of:

1. a set T of “terminals” *(usually denoted with lowercase letters)*
2. a set N of “non-terminals” *(usually denoted with uppercase letters)*
3. a unique “start symbol” $S \in N$
4. a set P of “productions” of the form $A \rightarrow \omega$, where $A \in N$ and $\omega \in (T+N)^*$

Notes:

- Two rules with the same left-hand side may be combined using an OR (“|”).

for example, the two rules:	can be combined into:
$A \rightarrow aBb$ $A \rightarrow Aa$	$A \rightarrow aBb \mid Aa$

- The language generated by the grammar G is denoted by L(G).

- Sometimes ::= is used in place of ➔. This is called Backus-Naur Form (BNF).

- A grammar describes the syntax rules for forming sentences.
  It tells us whether a particular string is “well-formed”, or “legal”.

- No satisfactory grammar has yet been developed for natural languages
  Natural languages are inherently *ambiguous* and *context-sensitive*.

An Example inspired by the English language:

<sentence>	➜	<noun-phrase> <predicate>
<noun-phrase>	➜	<article> <noun>
<predicate>	➜	<verb>
<article>	➜	the | a | an
<noun>	➜	cat | flower
<verb>	➜	jumps | blooms

“**the cat jumps**” and “**a flower blooms**” would be legal in the above grammar.

A grammar only defines what is syntactically correct, not necessarily what is meaningful. For example the above grammar will also generate sentences such as “**the cat blooms**”, and “**a flower jumps**” which of course do not make sense.

A grammar describes the syntax (the form) and *not* the semantics (the meaning). However, the syntax may, indirectly, affect the semantics (more on this later).

Two ways that grammars are used in compiling computer programs:

1. Given a grammar G and a string ω, determine if $\omega \in L(G)$. This is "parsing."
2. Given a description of language L, design a grammar G that generates L.

Example:

Consider the grammar G_1 with the following rules:

S ➔ aA
S ➔ AS
A ➔ *b*
A ➔ λ

Is the string **ba** $\in L(G_1)$? That is, is **ba** "legal" according to grammar G?

Answer: The string **ba** is legal only if it can be derived using G.
Start with S. Apply one rule at a time, replacing a *nonterminal* with the right side of an applicable rule, until only *terminals* remain.
A derivation of **ba** using grammar G is:

$$\mathbf{S} \Rrightarrow \mathbf{AS} \Rrightarrow \mathbf{AaA} \Rrightarrow \mathbf{baA} \Rrightarrow \mathbf{ba}$$

Example:

Build a grammar for all strings (of *a*'s & *b*'s) that begin with **a** and end with **b**.

Solution 1:

S ➔ aAb
A ➔ aA | bA | λ

Solution 2: (FSA style: generate one character at a time from left to right)

S ➔ aA // start with an a
A ➔ aA | bA | B // produce either a or b, repeatedly
B ➔ b // produce a final b and quit