Grammars

A grammar G is defined as a quadruple

\[ G = (V, T, S, P), \]

where

- \( V \) is a finite set of variables or non-terminal symbols,
- \( T \) is a finite set of terminal symbols, (alphabet)
- \( S \) in \( V \) is the start symbol, and
- \( P \) is a set of productions, or rewrite rules.
Productions have the form $x \rightarrow y$, where

- $x$ is a string of one or more terminal or non-terminal symbols, ($x$ is in $(V \cup T)^+$) and

- $y$ is a string of zero or more terminal or non-terminal symbols ($y$ is in $(V \cup T)^*$).
Example 1: This grammar generates all strings of properly nested brackets.

- \( V = \{ S \} \)
- \( T = \{ [, ] \} \)
- start symbol is \( S \)
- \( P: \)
  1. \( S \rightarrow [S]S \)
  2. \( S \rightarrow \epsilon \)
Example 2: This grammar generates all strings of the form $a^n b^n$, $n \geq 0$

- $V = \{ S \}$
- $T = \{ a, b \}$
- start symbol is $S$
- $P$:
  1. $S \rightarrow aSb$
  2. $S \rightarrow \epsilon$
Derivations. If $x \rightarrow y$ is a production, write

$$\alpha x \beta \rightarrow \alpha y \beta$$

to denote the application of rewrite rule $x \rightarrow y$.

Example 3. Derive the string $[]$ using the grammar in Example 1:

Example 4. Derive the string $[[]]$ using the same grammar.
Notation

- $\Rightarrow$ denotes a one step derivation
- $\Rightarrow^+$ denotes a derivation of one or more steps
- $\Rightarrow^*$ denotes a derivation of zero or more steps

Example 5. Compressing Example 4, we could write $S = \Rightarrow^* []$
A left(right)-most derivation is one in which the left(right)-most non-terminal is replaced at each step. The derivations in examples 3 and 4 are leftmost derivations.

Solving the problem in example 4 using a rightmost derivation, we have

\[ S \rightarrow [S] \rightarrow [S][S] \rightarrow [S][S] \rightarrow [S][] \rightarrow [] \]
If $S \Rightarrow^* \beta$, then $\beta$ is called a *sentential form*. In the item above, each of $S$, $[S]S$, $[S][S]S$, $[S][S]$, $[S]$, and $[]$ is a sentential form.
L(G), the language generated by grammar G, is the set of all strings of terminal symbols that can be obtained by starting with symbol S and applying rules from P.

\[ L(G) = \{ \alpha \in T^* | S \Rightarrow *\alpha \} \]

- For example 1, L(G) is the set of all properly nested square brackets
- For example 2, 
  \[ L(G) = \{ a^{*n}b^{*n}, \ n \geq 0 \} \].
BNF (Backus-Naur Form) was developed for the syntactic definition of Algol60. It is a type of grammar that differs only in notation from the grammars defined above.

BNF identifies non-terminal symbols by enclosing them in angle brackets, and uses the symbol ::= instead of -> in productions.
Here is a grammar that generates identifiers that consist of a letter followed by zero or more letters or digits:

- $V = \{ <\text{ident}>, <\text{letter}>, <\text{ident-tail}>, <\text{ident-elem}> \}$
- $T = \{ a, b, \ldots, z, 0, 1, \ldots, 9 \}$
- $<\text{ident}>$ is the start symbol
• $P$ is given by

1. $\text{<ident>} ::= \text{<letter>}\text{<ident-tail>}$
2. $\text{<letter>} ::= \text{a}|\text{b}|...|\text{z}$
3. $\text{<ident-tail>} ::= \text{<ident-elem>} \cdot \text{<ident-tail>}$
4. $\text{<ident-tail>} ::= \epsilon$
5. $\text{<ident-elem>} ::= \text{<letter>}$
6. $\text{<ident-elem>} ::= 0|1|...|9$

Exercise: derive x1.