Context-Free Grammars

A grammar is a set of rules for putting strings together and so corresponds to a language.

Grammars

A ***grammar*** consists of:

- a set of ***variables*** (also called nonterminals), one of which is designated the start variable; It is customary to use upper-case letters for variables;
- a set of ***terminals*** (from the alphabet); and
- a list of ***productions*** (also called rules).

Example: $\mathtt{0}^n\mathtt{1}^n$

Here is a grammar:

$$S \to \mathtt{0}S\mathtt{1}$$
$$S \to \varepsilon$$

S is the only variable. The terminals are $\mathtt{0}$ and $\mathtt{1}$. There are two productions.

Using a Grammar

A production allows one to take a string containing a variable and replace the variable by the RHS of the production.

String w of terminals is ***generated*** by the grammar if, starting with the start variable, one can apply productions and end up with w. The sequence of strings so obtained is a ***derivation*** of w.

We focus on a special version of grammars called a ***context-free grammar*** (CFG). A language is ***context-free*** if it is generated by a CFG.

Example Continued

$$S \to 0S1$$
$$S \to \varepsilon$$

The string 0011 is in the language generated. The derivation is:

$$S \Longrightarrow 0S1 \Longrightarrow 00S11 \Longrightarrow 0011$$

For compactness, we write

$$S \to 0S1 \mid \varepsilon$$

where the vertical bar means *or*.

Example: Palindromes

Let P be language of palindromes with alphabet $\{\mathtt{a}, \mathtt{b}\}$. One can determine a CFG for P by finding a recursive decomposition.

If we peel first and last symbols from a palindrome, what remains is a palindrome; and if we wrap a palindrome with the same symbol front and back, then it is still a palindrome.

CFG is

$$P \to \mathtt{a}P\mathtt{a} \mid \mathtt{b}P\mathtt{b} \mid \varepsilon$$

Actually, this generates only those of even length. . .

Formal Definition

One can provide a ***formal definition*** of a context-free grammar. It is a 4-tuple (V, Σ, S, P) where:

- V is a finite set of variables;
- Σ is a finite alphabet of terminals;
- S is the start variable; and
- P is the finite set of productions. Each production has the form $V \to (V \cup \Sigma)^*$.

Further Examples: Even 0's

A CFG for all binary strings with an even number of 0's.

Find the decomposition. If first symbol is 1, then even number of 0's remain. If first symbol is 0, then go to next 0; after that again an even number of 0's remain. This yields:

$$S \to 1S \mid 0A0S \mid \varepsilon$$
$$A \to 1A \mid \varepsilon$$

Alternate CFG for Even 0's

Here is another CFG for the same language.

Note that when first symbol is 0, what remains has odd number of 0's.

Alternate CFG for Even 0's

Here is another CFG for the same language.

Note that when first symbol is 0, what remains has odd number of 0's.

$$S \to 1S \mid 0T \mid \varepsilon$$
$$T \to 1T \mid 0S$$

Example

A CFG for the regular language corresponding to the RE 00^*11^*.

Example

A CFG for the regular language corresponding to the RE $\texttt{00}^*\texttt{11}^*$.

The language is the concatenation of two languages: all strings of zeroes with all strings of ones.

$$S \to CD$$
$$C \to \texttt{0}C \mid \texttt{0}$$
$$D \to \texttt{1}D \mid \texttt{1}$$

Example Complement

A CFG for the complement of RE $\texttt{00}^*\texttt{11}^*$.

CFGs don't do "and"s, but they do do "or's". A string *not* of the form $\texttt{0}^i\texttt{1}^j$ where $i, j > 0$ is one of the following: contains $\texttt{10}$; is only zeroes; or is only ones. This yields CFG:

$$
\begin{aligned}
S &\to A \mid B \mid C \\
A &\to D\texttt{10}D \\
D &\to \texttt{0}D \mid \texttt{1}D \mid \varepsilon \\
B &\to \texttt{0}B \mid \texttt{0} \\
C &\to \texttt{1}C \mid \texttt{1}
\end{aligned}
$$

Consistency and Completeness

Note that to check a grammar and description match, one must check two things: that everything the grammar generates fits the description (***consistency***), and everything in the description is generated by the grammar (***completeness***).

Example

Consider the CFG

$$S \to 0S1S \mid 1S0S \mid \varepsilon$$

The string 011100 is generated:

$$S \Longrightarrow 0S1S \Longrightarrow 01S \Longrightarrow 011S0S \Longrightarrow 0111S0S0S$$
$$\Longrightarrow 01110S0S \Longrightarrow 011100S \Longrightarrow 011100$$

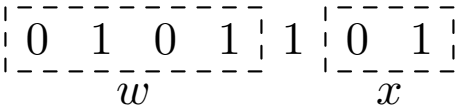
What does this language contain? Certainly every string generated has equal 0's and 1's...

But can any string with equal 0's and 1's be generated?

Example Argument for Completeness

Yes. All strings with equal 0's & 1's are generated:

Well, at some point, equality between 0's and 1's is reached. The key is that if string starts with 0, then equality is first reached at a 1. So the portion between first 0 and this 1 is itself an example of equality, as is the portion after this 1. That is, one can break up string as $0w1x$ with both w and x in the language.

The break-up of 00101101: 

A Silly Language CFG

This CFG generates sentences as composed of noun- and verb-phrases:

$S \rightarrow NP\ VP$
$NP \rightarrow \texttt{the}\ N$
$VP \rightarrow V\ NP$
$V \rightarrow \texttt{sings} \mid \texttt{eats}$
$N \rightarrow \texttt{cat} \mid \texttt{song} \mid \texttt{canary}$

This generates "the canary sings the song", but also "the song eats the cat".

This CFG generates all "legal" sentences, not just meaningful ones.

Practice

Give grammars for the following two languages:

1. All binary strings with both an even number of zeroes and an even number of ones.
2. All strings of the form $\texttt{0}^a\texttt{1}^b\texttt{0}^c$ where $a + c = b$. (Hint: it's the concatenation of two simpler languages.)

Practice Solutions

1)

$S \to \mathtt{0}X \mid \mathtt{1}Y \mid \varepsilon$

$X \to \mathtt{0}S \mid \mathtt{1}Z$ (odd zeroes, even ones)

$Y \to \mathtt{1}S \mid \mathtt{0}Z$ (odd ones, even zeroes)

$Z \to \mathtt{0}Y \mid \mathtt{1}X$ (odd ones, odd zeroes)

2)

$S \to TU$

$T \to \mathtt{0}T\mathtt{1} \mid \varepsilon$

$U \to \mathtt{1}U\mathtt{0} \mid \varepsilon$

Summary

A context-free grammar (CFG) consists of a set of productions that you use to replace a variable by a string of variables and terminals. The language of a grammar is the set of strings it generates. A language is context-free if there is a CFG for it.