Context-free languages

Def: A context-free grammar is a grammar where all the productions are of the form

\[ A \rightarrow \text{single variable} \]

Def: A language \( L \) is context-free if there is a context-free grammar generating \( L \).

(In a week and a half we'll have automatons for these things - we start with grammars.)

E.g. \( L = \{ a^n b^n \mid n \in \mathbb{Z} \} \)

Here is a CFG for \( L \):

\[ S \rightarrow aSb | \lambda \quad (S \text{ is the only variable}) \]
I can produce \( a^3 b^3 \) using this grammar by

\[
S \rightarrow a S b \rightarrow a a S b b \rightarrow a a a S b b b b \rightarrow a a a b b b b .
\]

**E.g.** \( L = \{ \text{strings of } (, ) \text{ that make sense as a matching set of parens - reading } L \text{ to } R, \# ('s \geq \#)'s always, and \# ('s = \#)'s at the end} \} \)

Grammar for this

\[
S \rightarrow \lambda | (S) | SS
\]

**E.g.** \( L = \{ a^k b^k c^n \} \)

\[
S \rightarrow a S T
\]

\[
T \rightarrow b T c | \lambda
\]

To make \( a a b b c c c c c \), we do

\[
S \rightarrow a S \rightarrow a a S \rightarrow a a a T \rightarrow a a b T c \rightarrow a a b b T c c \rightarrow a a b b b T c c c
\]
Derivations for getting strings out of a CFG:

E.g. \[ S \rightarrow \varepsilon | (S) | SS \]

to generate \((())()\), we could do:

\[ S \rightarrow SS \rightarrow S(S) \rightarrow (S)(S) \]
\[ \rightarrow (S)(()) \rightarrow ((S))(() \rightarrow (()()) \]

or we could do

\[ S \rightarrow SS \rightarrow (S)S \rightarrow ((S))S \]
\[ ((S))(S) \rightarrow (())S \rightarrow ()() \]

We did the same moves in the 2 processes-
we just did them in a different order.
How do we say this in a precise way?
We can draw the derivation tree for this derivation:
Given a derivation tree, there is always a leftmost derivation - you always replace the leftmost remaining variable first, so the leftmost derivation for this tree is

\[ S \rightarrow SS \rightarrow (S)S \rightarrow ((S))S \rightarrow (((S)))(S) \rightarrow (((S)))(S). \]
\[ L = \{ \text{ww}^R \mid w \in \{a, b\}^* \} \]

\[ w \text{ backwards} \]

grammar given by:

\[ S \rightarrow aSa | bSb | \lambda \]

\[ L = \{ \text{ww}^R \mid w \in \{a, b\}^* \} \]

grammar given by:

\[ S \rightarrow aSa | bSb | a | b | \lambda \]

\[ L = \{ \text{ww}^R \mid \#a's > \#b's \} \]

\[ S \rightarrow aSa | bSb | a | b | \lambda \]

hope - can't get aababbbbaa (using SS requires 2 extra a's)

\[ S \rightarrow aTa | Ta | T \]

\[ T \rightarrow aTb | bTa | TT \]

\[ \]
\[ L = \{ \omega \mid \#a's = \#b's \} \]

\[ S \rightarrow asb | bsa | abs | bas | \varepsilon \]

Example strings:

- ab, ba
- abba,
- baab, abab,
- baba,
- aababbbba

They aren't wrong, just now redundant.
To allow arbitrary # of extra $a$'s

$S \rightarrow aT \mid Ta \mid TaT$

$T \rightarrow aTb \mid bTa \mid TT \mid \lambda a$

Or:

$S \rightarrow aT \mid Ta \mid TaT \mid SS$

$T \rightarrow aTb \mid bTa \mid TT \mid \lambda$