Context-free languages

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

$$A \longrightarrow \underline{\qquad\qquad}$$

single variable

Def: A language L is context free if there context-free grammar generating L.

(In a week and a half we'll have automata for these things - we start w/ grammars.)

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

Here is a CFG for L:

$$S \longrightarrow aSb \mid \lambda$$ (S is the only variable)

I can produce a^3b^3 using this grammar by

$$S \to aSb \to aaSbb \to aaaSbbb$$

$$\to aaabbb.$$

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

Grammar for this

$$S \to \lambda \mid (S) \mid SS$$

E.g. $L = \{a^k b^n c^n\}$

$$S \to aS \mid T$$

$$T \to bTc \mid \lambda$$

To make aabbbccc, we do

$$S \to aS \to aaS \to aaT \to aabTc \to aabbTcc \to aabbbTccc$$

aabbbccc

Derivations for getting strings out of a CFG:

E.g. $S \to \lambda \mid (S) \mid SS$

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

$$S \to SS \to S(S) \to (S)(S)$$
$$\to (S)() \to ((S))() \to (())()$$

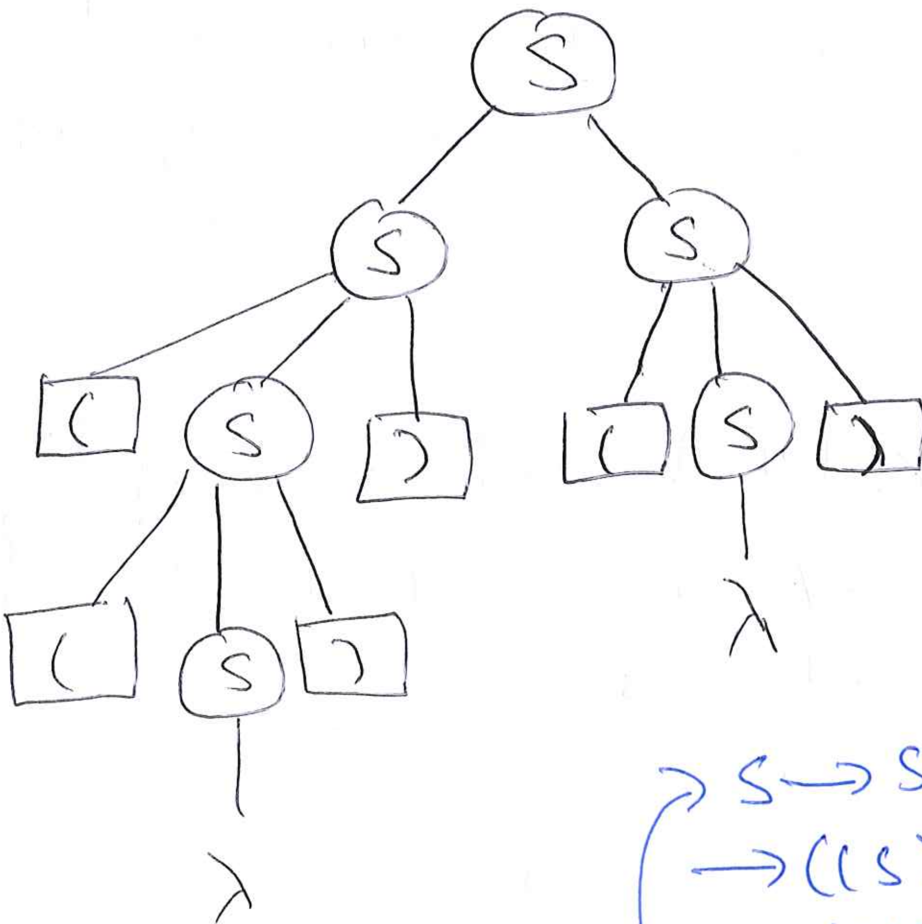
or we could do

$$S \to SS \to (S)S \to ((S))S$$
$$((S))(S) \to (())\cancel{(S)} \to (())().$$

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 <u>derivation tree</u> for this derivation:



Given a derivation tree, there is always a <u>leftmost derivation</u> - you always replace the leftmost remaining variable first, so the leftmost derivation for this tree is

$$S \to SS \to (S)S \to ((S))S \to (())S \to (())(S) \to (())().$$

$$L = \{ \underbrace{w w^R}_{\text{w backwards}} \mid w \in \{a,b\}^* \}$$

grammar given by:

$$S \rightarrow aSa \mid bSb \mid \lambda$$

$$L = \{ w \in \{a,b\}^* \mid w = w^R \}$$

grammar given by:

$$S \rightarrow aSa \mid bSb \mid a \mid b \mid \lambda$$

$$L = \{ \cancel{\#a's}\ w \mid \#a's > \#b's \}$$

$$S \rightarrow aSb \mid bSa \mid abS \mid baS \mid SS \mid a$$

nope - can't get aababbbaa
(using SS requires 2 extra a's)

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

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

This would give strings w/ one extra a.

$L = \{ w \mid \#a\text{'s} = \#b\text{'s} \}$

$S \rightarrow aSb \mid bSa \mid \cancel{abS} \mid \cancel{baS}$
$\quad SS \mid \lambda$

they aren't wrong, just now redundant.

Example strings:

ab, ba

abba,

baab, abab,

baba,

aababbba.

To allow arbitrary # of extra a's

$S \to aT \mid Ta \mid TaT$

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

Or:

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

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