Context-free languages

9/28/18

So far, we have been talking about regular languages:

3 equivalent characterizations

a) L is recognized (the set accepted by) by a DFA/NFA

b) L is the set of strings matching a regular expression

c) L is the set generated by a (right)-regular grammar.

We saw this week that many languages are not regular.

We want a bigger class of languages.

Our next class of languages:

Context-free languages.

Right now: grammars for CFLs which are called CFGs. (context-free grammars)

Week after next: Pushdown automata (the equiv. of DFAs for these langs

~~Def~~: A ~~lang~~ grammar G is context-free if every production is of the form

$$A \to w,$$

where $A \in V$, $w \in (V \cup \Sigma)^*$

set of variables
alphabet
start var
productio

(More formally, if $G = (V, \Sigma, S, P)$, G is context free if $P \subseteq V \times (V \cup \Sigma)^*$.)

(Generally we only require $P \subseteq ((V \cup \Sigma)^* \setminus \{\lambda\}) \times (V \cup \Sigma)^*$.)

Examples:

$L = \{a^n b^n \mid n \in \mathbb{Z}\} = \{$ ~~n a's~~ some # of a's followed by same # of b's $\}$

Grammar generating ~~th~~ L:

$V = \{S\}$, $\Sigma = \{a, b\}$, $S = S$,

$P = \{ \; S \to aSb \mid \lambda \; \}$

What can we generate?

$$S \to \lambda,$$

$$S \to aSb \to ab,$$

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

$$\vdots$$

<u>Example</u>: Valid parenthesizations:

strings of ( ), equal # of both, must see at least ~~more~~ as many (~~than~~ as) at any point.

$$V = \{S\}, \quad \Sigma = \{ (,) \}, \quad S = S$$

$$P = \{ S \to (S) \mid \lambda \mid SS \}$$

Does this get everything? I think so, but try something medium-sized & random:

(()(())(()(())))

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

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

$\cancel{\to (()(S}\quad (()\underline{S}\,\underline{S})$

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

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

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

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

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

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

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

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

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

$\to (()(())(()(())))$

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

A CFG:

$$S \longrightarrow aSa \mid bSb \mid \lambda$$

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

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

$L = \{a^{n^2} \mid n \in \mathbb{Z}\}$

actually, this isn't CF

$L = \{w \in \{a,b\}^* \mid \#a\text{'s} = \#b\text{'s}\}$

$$S \longrightarrow aSb \mid bSa \mid \lambda \mid SS$$

This gets λ, ab, ba, aabb, ~~ab~~

Can we get abba?

$$S \rightarrow aSb \longrightarrow abSab$$

Now we can:

$$S \to SS \to aSbS \to abS \to abbSa \to abba.$$

Note: we have 2 ways to get $abab$

$$S \to SS \to aSbS \to abS \to abaSb \to abab$$

or $S \to aSb \to abSab \to abab$.

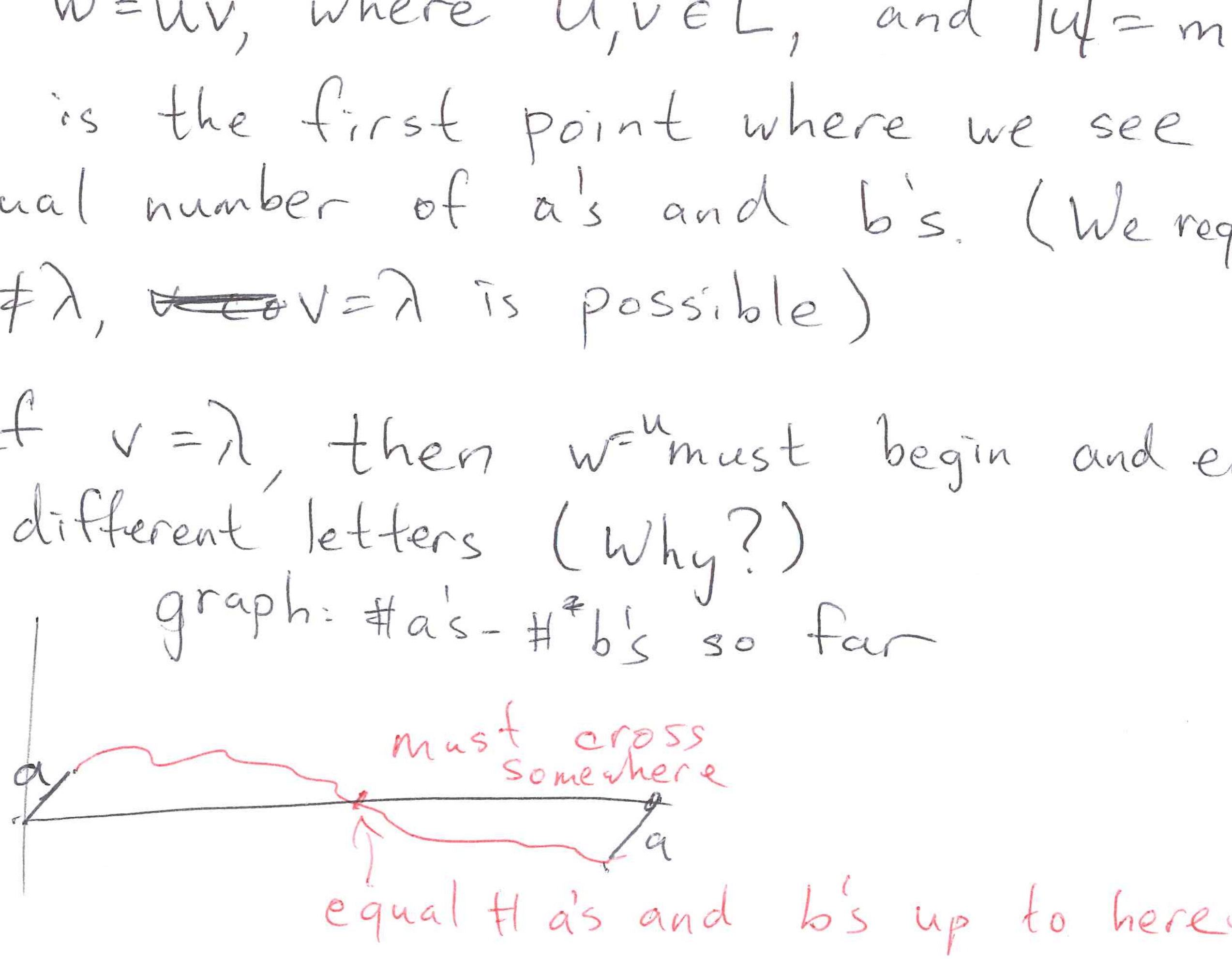
Possible proof this works:

Given $w \in L$, we can write

$w = uv$, where $u, v \in L$, and $|u| = m$,
m is the first point where we see equal number of a's and b's. (We require $u \neq \lambda$, ~~v ≠~~ $v = \lambda$ is possible)

If $v = \lambda$, then $w = u$ must begin and end w/ different letters (Why?)

graph: #a's - #b's so far



So if w began and ended w/ same letter, $v \neq \lambda$.

When w begins and ends w/ different letters, $w = axb$ or $w = bxa$, where $x \in L$. By induction, we can generate x from S, so we can gen w by doing $S \to aSb \to \cancel{a} \cdots \to axb$

or $S \to bSa \to \cdots \to bxa$.

Otherwise $w = uv$, $u, v \in L \setminus \{\lambda\}$, so use $S \to SS \to \cancel{uv} \cdots \to uv$

Induction