Closure properties of context-free languages

**Facts:** If $L_1 + L_2$ are context-free languages,

1) $L_1 \cup L_2$ is a context-free lang.

   (*Pf:* Given grammars for $L_1 + L_2$ w/ start vars $S_1, S_2$, create a new grammar w/ start var $S$ and prod. $S \rightarrow S_1 | S_2$ along w/ all old prods. This new grammar generates $L_1 \cup L_2$)

2) $L_1L_2$ is a context-free lang.

   (*Pf:* $S \rightarrow S_1S_2$)

3) $L^*$ is context-free

   (*Pf:* $S \rightarrow S_1S_1^*$)
4) $L_1 \cap L_2$ might not be a context-free language.

Eg. $L_1 = \{ a^n b^n c^k \mid n \geq 0, k \geq 0 \}$
$L_2 = \{ a^k b^n c^n \mid n \geq 0, k \geq 0 \}$

are both context-free, but
$L_1 \cap L_2 = \{ a^n b^n c^n \mid n \geq 0 \}$

is not.

Intuition— the reg. lang. proof (where you have a DFA that simulates running 2 DFAs simultaneously) doesn't work b/c you need 2 stacks to run 2 PPAs simultaneously.

But:

4') If $L_1$ is context-free and $L_2$ is regular, $L_1 \cap L_2$ is context-free.

(running a PPA + a DFA simultaneously only requires one stack)
5. $L_1$ might not be context-free either.

Intuition: nondeterminism adds power to PPs. (Our reg. lang. pf only worked on DFAs and relied on converting NFAs to DFAs)

E.g. We now easily know that:

- $L_1 = \{ abw | n \geq 0, \text{l}(w) = 5 \}$ is context-free
- $L_2 = \{ \text{concatenation of a CF lang w/regex lang} \}$
- $L_3 = \{ a^i b^i n | n \geq 0 \}$ is context-free
- $L_4 = \{ \text{intersection of a CF lang w/regex lang} \}$
- $L_5 = \{ a^i b^j \text{ where } i \neq j \}$ does not divide $L_3$
- $L_6 = \{ a^i b^k \text{ where } k \neq 3 \}$ is context-free.
More HW 4 problems:

Algorithm to decide if $L = \Sigma^*$
(for $L$ regular)

Given a DFA $M$

... (You need something more clever than testing one string at a time, b/c that will take an infinite amount of time.)

For every string to be accepted, every reachable state has to be final.

Algorithm to decide if $L$ has infinitely many even length strings.

"Cheap sol'n": $K = \{ v \in \Sigma^* \mid |v| \text{ is even} \}$

$K$ is regular b/c

Given a machine for $L$, we can build a machine for $L \cap K$-check if that has reachable loops...
Expensive sol'n that you have to think about. Check if machine for $L$ has any loops, but even length loops starting an odd distance from initial state don't count.

A pumping lemma problem:

$L = \{a^n b^l | n/l \text{ is an integer} \}$

Given $m$, let $w = a^m b^m$. Then let $w = xyz$, $|xy| \leq m$, $y \neq \lambda$. We know from the restrictions $y = a^j$ for some $j$, $1 \leq j \leq m$. Then $xy^2z = a^{m+j} b^m$ bad if $j = m$ (which is possible)

$xyz = a^{m-j} b^m$ also bad if $j = m$

$xy^3z = a^{m+2j} b^m$ also bad if $j = m$

We don't want to allow adding/subtracting as many a's as there are b's. So if we have $(m+1)$ b's...
Given \( m \), let \( w = a^m b^{m+1} \). Let \( w = xyz \), \( |xy| \leq m \), \( y \neq \lambda \). From the restrictions \( y = a^j \) for some \( j \), \( 1 \leq j \leq m \).

Then \( xy^0z = a^{m+j} b^{m+1} \).

Since \( 1 \leq j \leq m \), \( m+1-j < m+1 \),

and \( m+1-j > 0 \), hence \( \frac{m+1-j}{m+1} \) is not an integer, so

\( xy^0z \notin L \).

Hence, \( L \) violates the pumping lemma and is not regular.