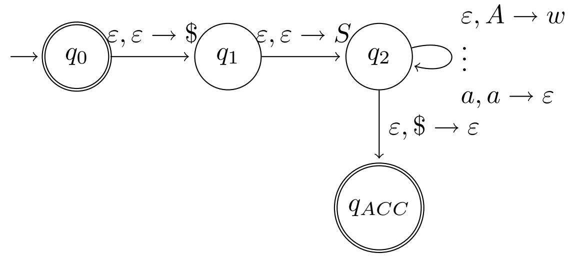
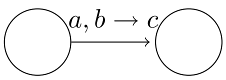
Notes for Monday, May 17th

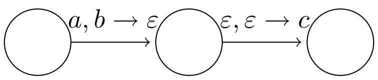
(Notes supplementary to slides) Contstruction of a CFG to PDA follows this construction:



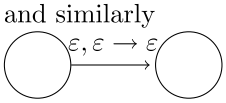
In PDA to CFG proof, we can get the PDA into the required form by splitting things of the form



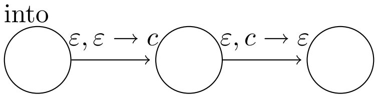
Into



and similarly



into



(diverging from the slide discussion now)

Consider the example $L = \{0^n 1^n 0^n | n \geq 0\}$. Can we make a CFG or PDA for L? If we had two stacks it would be easy. The attempted grammar

$$S \to 0A1B0|\varepsilon$$
$$A \to 0A1|\varepsilon$$
$$B \to 1B0|\varepsilon$$

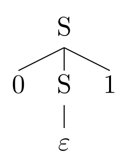
doesn't work. It turns out that L is not a CFL. To prove this, we'd need a pumping lemma for CFLs!

An example of how this pumping lemma would work: $L = \{0^n 1^n | n \geq 0\}$. We have already discussed the grammar for this in class:

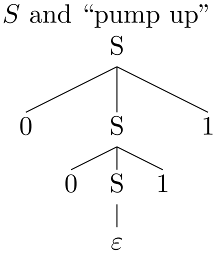
$$S \to 0S1|\varepsilon$$

It has one variable, S. So, $|V| = 1$. A parse tree of height 2 would have the longest path have three nodes (with a leaf as a terminal), so there are two variables in the path. Since there is only one variable on V, the path will repeat a variable due to pigeonhole. Hence, we can change the size of the tree!

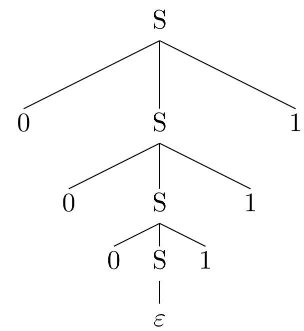
For the string $w = 01$, the tree is



We may replace the bottom S by the top S and "pump up" to get the string $0^2 1^2$...



...or again for $0^3 1^3$:



and so on. We may also "pump down" by replacing the upper S by the lower S to get the string $0^0 1^0 = \varepsilon$:

This outlines how the pumping lemma for CFLs works. Next time, a formal proof is given.