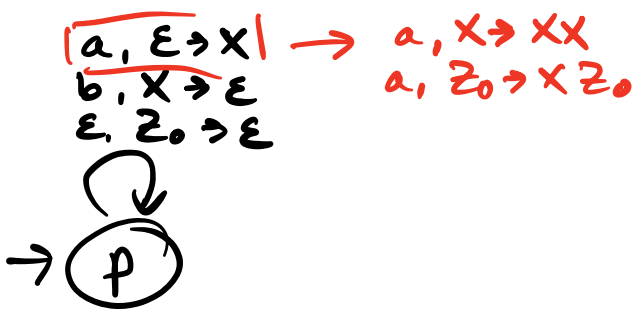
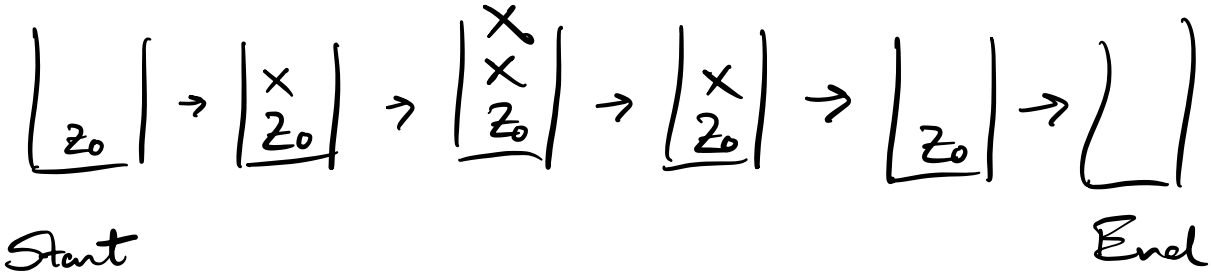
Let's now show that $L_{PDA} \subseteq L_{CFG}$

Ex Consider following NPDA which accepts via empty stack



On input $w = aabb$

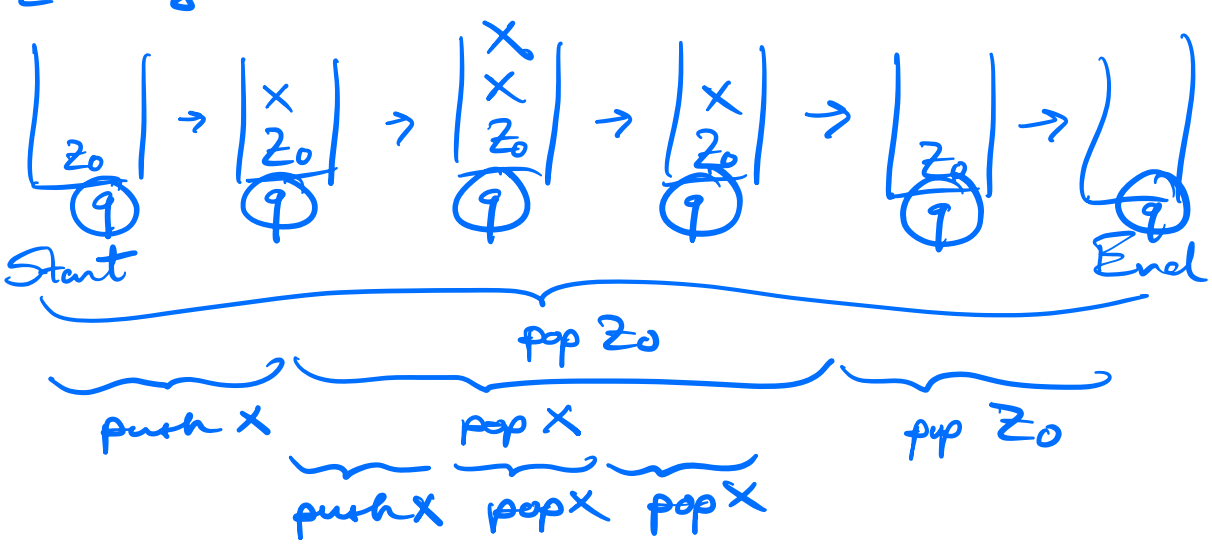


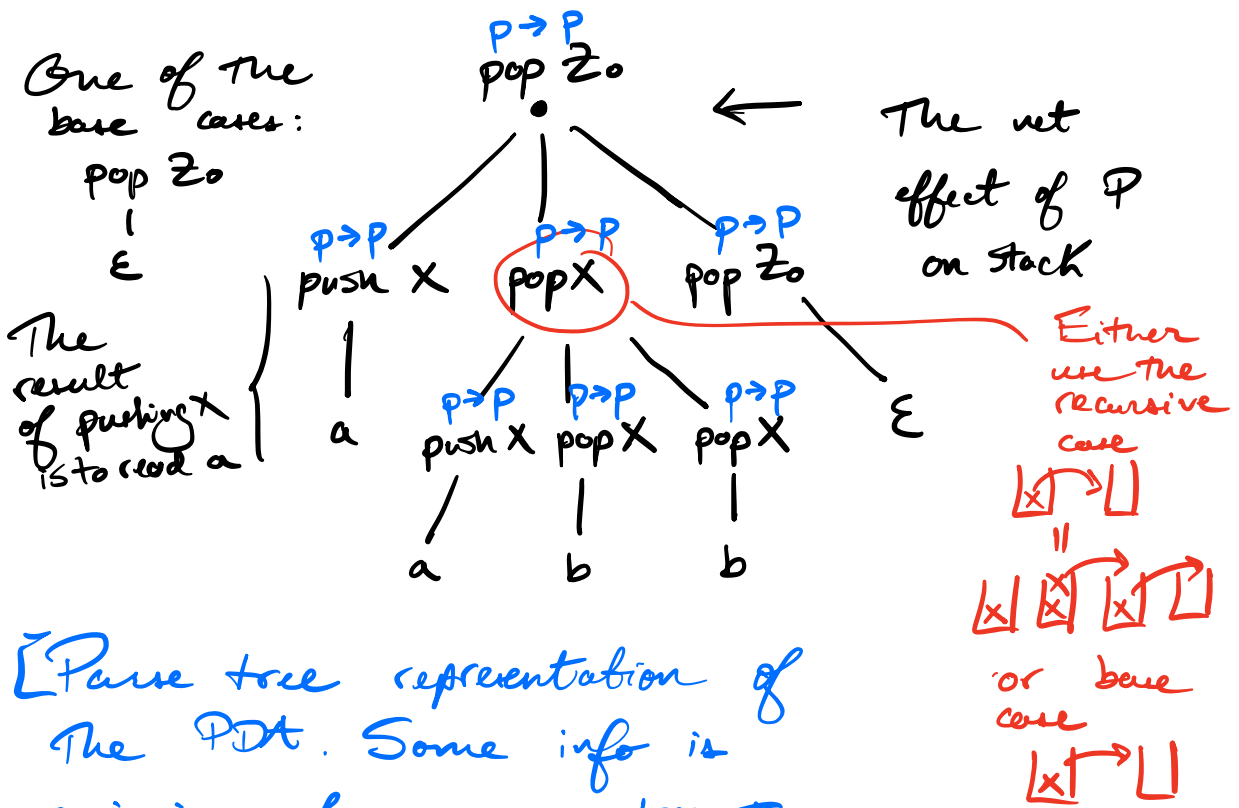
Conversion to CFG

Let's take the same computation & represent it as a parse tree Why? We're trying to get the equivalent CFG! [We will use the internal nodes to represent the machine pushing / popping and the leaf nodes to represent what letters (Terminals) are read (generated) during this process.]

Effectively, we represent the PDA's push/pop recursively starting from pop Z_0 at the top level.

[Copy & paste above tree]





[Parse tree representation of the PDA. Some info is missing from parse tree to accurately represent P's computation. What is it? The states! Need this otherwise base cases would change.]

We're ready to propose a grammar based on P that could generate this parse tree:

A variable

$S \to [\text{pop } Z_0, p \to p]$ (Top level)

$[\text{pop } Z_0, p \to p] \to \varepsilon$ (Base case # 1)

$[\text{pop } X, p \to p] \to b$ —"— # 2

$[\text{push } X, p \to p] \to a$ —"— # 3

$[pop\ Z_0, p \to p] \to [push\ X, p \to p]\cdot$
$[pop\ X, p \to p]\cdot$
$[pop\ Z_0, p \to p]$ (Rec case #1)

$[pop\ X, p \to p] \to [push\ X, p \to p]\cdot$
$[pop\ X, p \to p]\cdot$
$[pop\ X, p \to p]$ (—"— #2)

Let's simplify notation a little by

1. Using $[pop\ Z_0, p \to p] = p Z_0 p$ as the var name.
2. B/c now with 1. can't distinguish push vs pop, place the base case of pushing directly in the popping recursive case. $p Z_0 p \to a\ p X p\ p Z_0 p$

<u>The corresponding grammar is:</u>

$S \to p Z_0 p$
$p Z_0 p \to \varepsilon \mid a\ p X p\ p Z_0 p$
$p X p \to b \mid a\ p X p\ p Z_0 p$

<u>For you:</u> What happens when have $|G| > 1$?

Upshot

Despite their difference in appearance,

$$L_{CFG} = L_{NPDA \text{ via } \sqcup} = L_{NPDA \text{ via } \%}$$