How to prove a CFG is correct for a L.

Eg G: $S \to aSb \mid \varepsilon$ $\qquad A^nB^n = \{a^n b^n \mid n \geq 0\}$

Claim: $L(G) = A^nB^n$.

Proof: $\boxed{\text{I. } L(G) \subseteq A^nB^n}$

i.e. we must show that any w generated in G is of the form $a^i b^i$ for some i.

Let w be any string generated in G.

Then either

1. $w = \varepsilon$, ie. $w = a^0 b^0$, or
2. $w = \underline{a}\, w' \,\underline{b}$ where w' is a $\underline{\text{smaller}}$ string generated from S

   Then by the Ind.Hyp., w' is of the form $a^i b^i$ for some i

   Hence $w = a \cdot a^i b^i b = a^{i+1} b^{i+1}$, and $w \in A^nB^n$.

or, more traditionally:

Claim: $\forall\, w \in L(G)$, $w = a^i b^i$ for some i.

Proof: By induction on the number of steps in a shortest leftmost derivation of w.

Basis: Number of steps is 1

ie $w = \varepsilon$. ie $w = a^0 b^0$ $\boxtimes$

Induction: Let w be any string derivable in K steps, $K > 1$,

Ind Hyp: $\forall\, w'$ where w' is derivable in $< K$ steps, $w' = a^i b^i$ for some i.

Then $w = a w' b$. $\therefore$ w' is derivable in $< K$ steps. Then by the Ind Hyp, $w' = a^j b^j$ for some j.

$\therefore$ $w = a^{j+1} b^{j+1}$. $\boxtimes$

$S \to aSb \mid \varepsilon$.
(1) (2)

II. $A^nB^n \subseteq L(G)$.

Let $w \in A^nB^n$. Then $w = a^i b^i$ for some i.
Consider the following derivation of w in G:

$$\underbrace{S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow \ldots\ldots \Rightarrow a^iSb^i}_{\text{number of applications of rule (1) is } i} \Rightarrow a^ib^i.$$

∎

Eg. $L_1 = \{ w \in \{a,b\}^* \mid \#_a(w) = \#_b(w) \}$.

A grammar G_1 for L_1: $S \to aSb \mid bSa \mid SS \mid \varepsilon$
(2) (3) (4) (1)

Claim: $L(G_1) = L_1$.

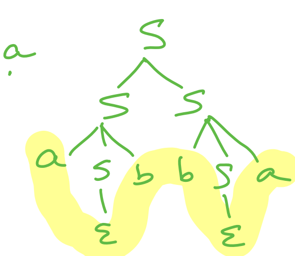
Proof: $L(G_1) \subseteq L_1$ ie $\forall w \in L(G)$, w has "$\#_a = \#_b$"

1. ε has 0 a's, 0 b's ie has "$\#_a = \#_b$"
2. $aw'b$ if w' has "$\#_a = \#_b$", then so does $aw'b$
3. $bw'a$ " w' " " " " $bw'a$.
4. w_1w_2 if w_1 and w_2 have "$\#_a = \#_b$" then so does w_1w_2

∎

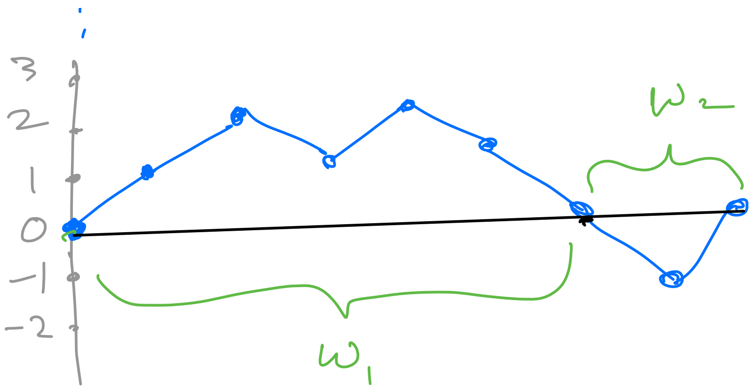
$S \to aSb \mid bSa \mid SS \mid \varepsilon$

abba

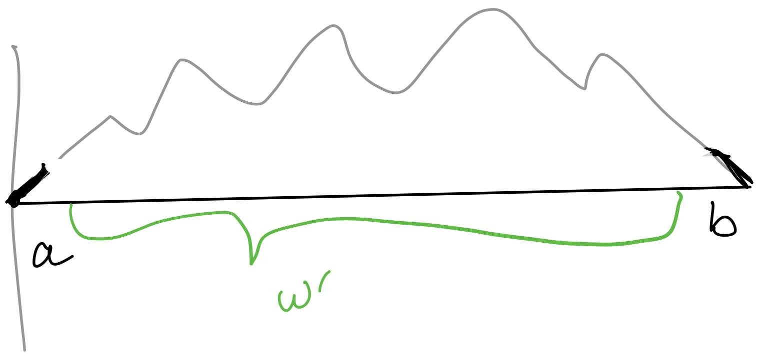


Part 2: $L_1 \subseteq L(G_1)$.
Let w be any string with $\#_a = \#_b$.
Let's graph the "excess a's" in the the prefixes of w.

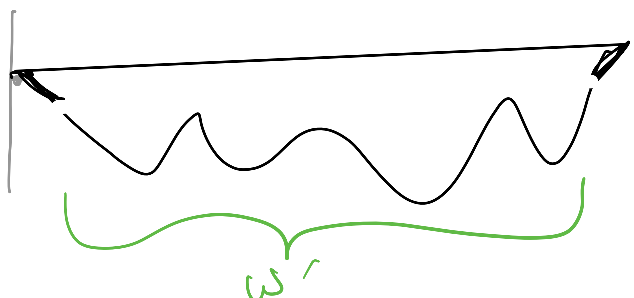
! a a b a b b b a



$S \Rightarrow SS \Rightarrow \dots$



$S \Rightarrow aSb \Rightarrow \dots$

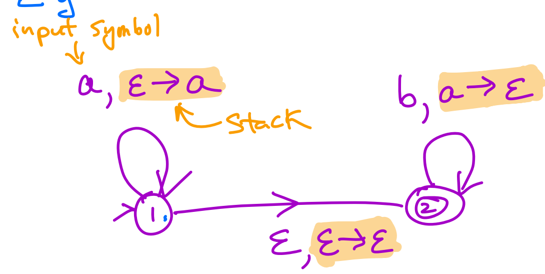


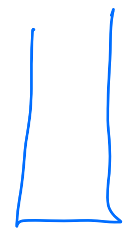
$S \Rightarrow bSw \Rightarrow \dots$

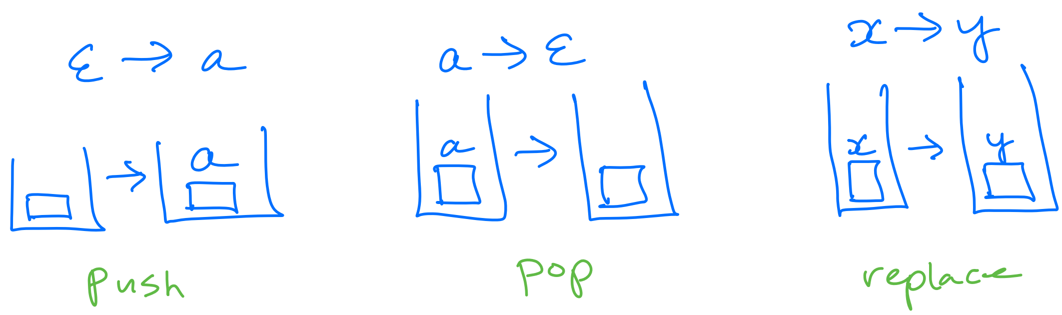
Section 2.2 Push-Down Automata (PDAs)

- Like an NFA with a stack
- transition: in a state, can read – an input symbol
  can read & pop what's on top of stack.
  then decide – new state to go to
  – push a new symbol on stack.
- A stack is LIFO. store of symbols
  – can be arbitrarily large.

Eg a PDA for A^nB^n: aabb







Definition 2.13 A **Push-down Automaton (PDA)** is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where:

- Q is a finite set of states
- Σ is input alphabet
- Γ is a stack alphabet
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \underline{\mathcal{P}(Q \times \Gamma_\varepsilon)}$ is transition function
- $q_0 \in Q$ is start state
- $F \subseteq Q$ is set of accept states.

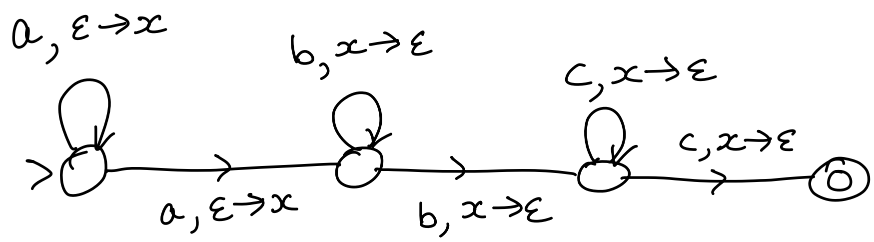
Σ_ε is $\Sigma \cup \{\varepsilon\}$ $\quad \Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$.

Note: the definition of PDA includes non-determinism.

Defn: A string w is <u>accepted</u> by a PDA M if $\exists$ a sequence of transitions M can make on input w that leads to a final state & $\boxed{\text{empty stack}}$ & (no input) accept

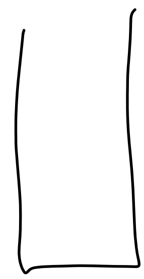
Eg $\{a^{i+j} b^i c^j \mid i, j \geq 1\}$

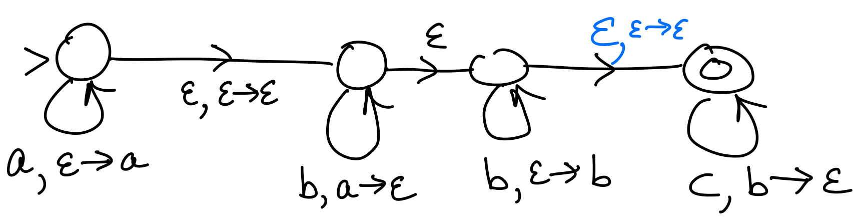
$S \to aSc \mid aXc$
$X \to aXb \mid ab$

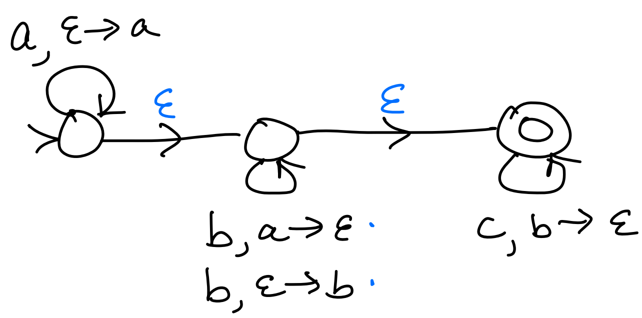


Eg $\{a^i b^{i+j} c^j \mid i, j \geq 0\}$.

aaa bbbb c







aabbbc