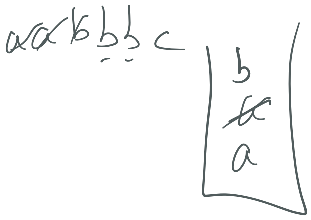
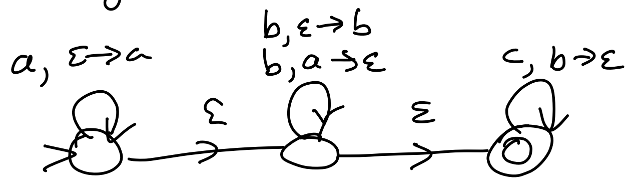
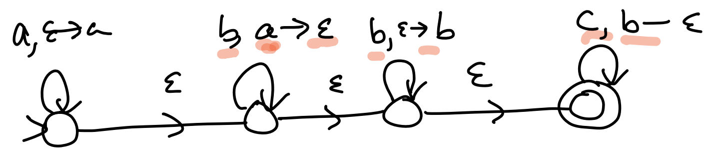
Recall from last time $\{ a^i b^{i+j} c^j \mid i, j \geq 0 \}$. ← Find a CFG.

Is that language inherently non-deterministic?





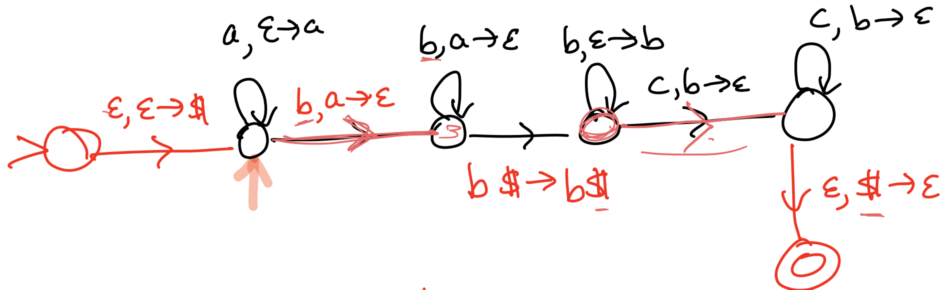
⇓ more deterministic.



Note: Your text always use bottom-of-stack marker.

Sipser's defⁿ of PDA accepting a string is a little different but equivalent.

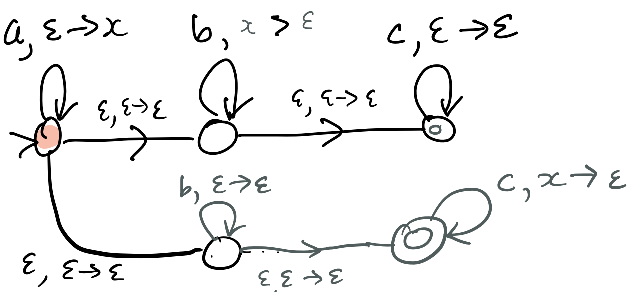
⇓ more deterministic: detect bottom-of-stack.



deterministic

Can we always remove the non-determinism?

$\{ a^i b^j c^k \mid i = j \text{ or } i = k \}$.



This PDA has no deterministic equivalent.

We may also want to detect end-of-string
end-of-input.

Then we say, instead of finding a PDA that recognizes L, we find a PDA for $L\#$.

"end of string" symbol at end of each string in $L\#$

Theorem 2.20

A language is CF iff $\exists$ a PDA that recognizes it.

ie $\exists$ a CFG that generates it

ie $\exists$ a CFG in CNF that generates it

Proof I ($\Rightarrow$) S'pose L is CF ie has a CFG G_L.

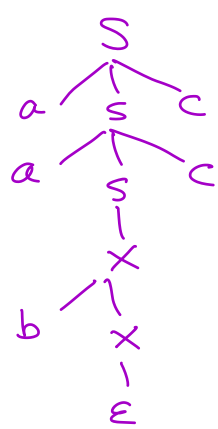
[want to show: $\exists$ a PDA that recognizes L]

We will construct a PDA that simulates deriving a string in the grammar G_L.

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

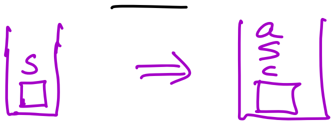
Eg $S \to aSc \mid X$
$X \to bX \mid \varepsilon$

aabcc

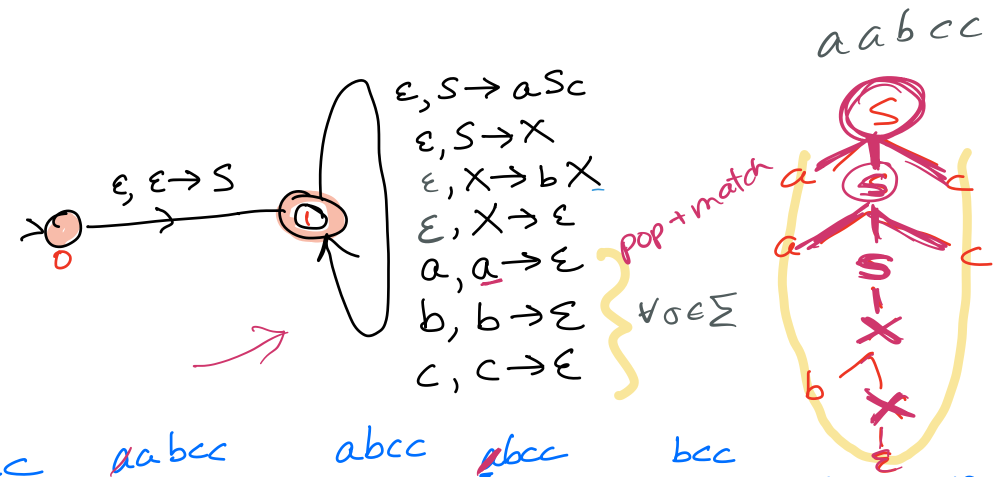


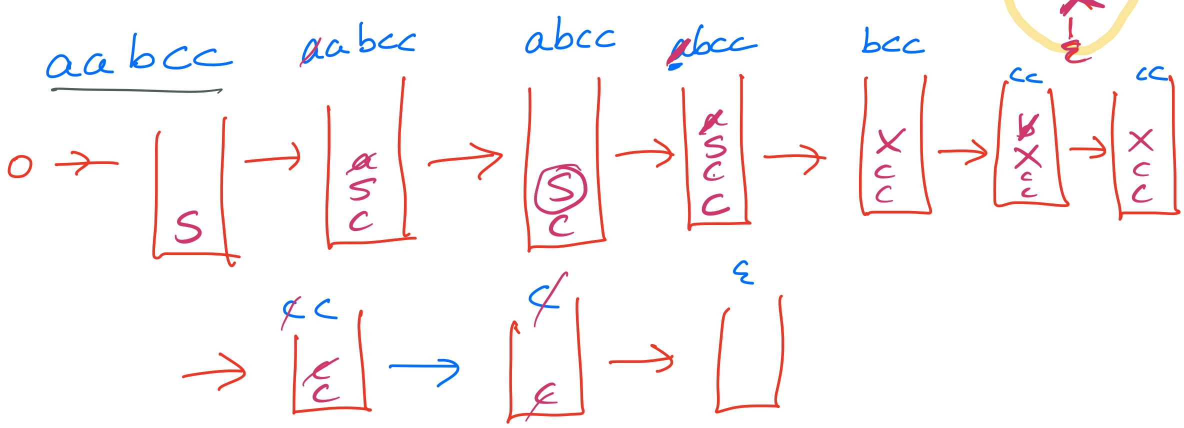
$S \Rightarrow aSc \Rightarrow aaScc \Rightarrow aaXcc \Rightarrow aabXcc$
$\Rightarrow aabcc$

How the stack will be used.



Top-Down Parser for $S \to aSc \mid X$, $X \to bX \mid \varepsilon$.





In general

$\forall$ CFG $G = (V, \Sigma, R, S)$ $\exists$ a Top-Down Parser PDA M that recognizes $L(G)$ where

$$M = (\{q_0, f\}, \Sigma, \Sigma \cup V, \delta, q_0, \{f\})$$

and where the transitions of δ are as follows:

$$\delta(q_0, \varepsilon, \varepsilon) = \{(f, S)\}$$

$$\forall \sigma \in \Sigma, \quad \delta(f, \sigma, \sigma) = \{(f, \varepsilon)\}$$

Feb 9

$$\forall Z \in V, \quad \delta(f, \varepsilon, Z) = \bigcup_{Z \to \gamma} \{(f, \gamma)\}$$

rules in R.

*.

II ($\Leftarrow$) $\forall$ PDA M, $\exists$ CFG G s.t. M recognizes $L(G)$

First, simplify M so that:

1. it has a single accept state.
2. each transition EITHER — pushes a symbol / Not both.
   — pops a symbol

What we are interested in doing is creating variables

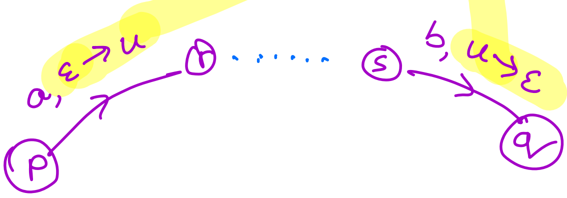
A_{pq} — has the property that $\exists$ a derivation

$A_{pq} \Rightarrow \cdots \Rightarrow \underline{abbccb}$ ← an example string of Σ^*

iff $\exists$ a way M can go from state p to state q consuming input abbccb

net result on stack is ε

$\forall$ pairs of states p, q



then $A_{pq} \to a A_{rs} b$

Remember:

A_{pq} "means"

"strings that can take M from p to q."

$$A_{pq} \to A_{pr} A_{rq}$$

$$A_{pp} \to \varepsilon$$

Start symbol is : $A_{q_0 q_{accept}}$ $\blacksquare$