Announcements

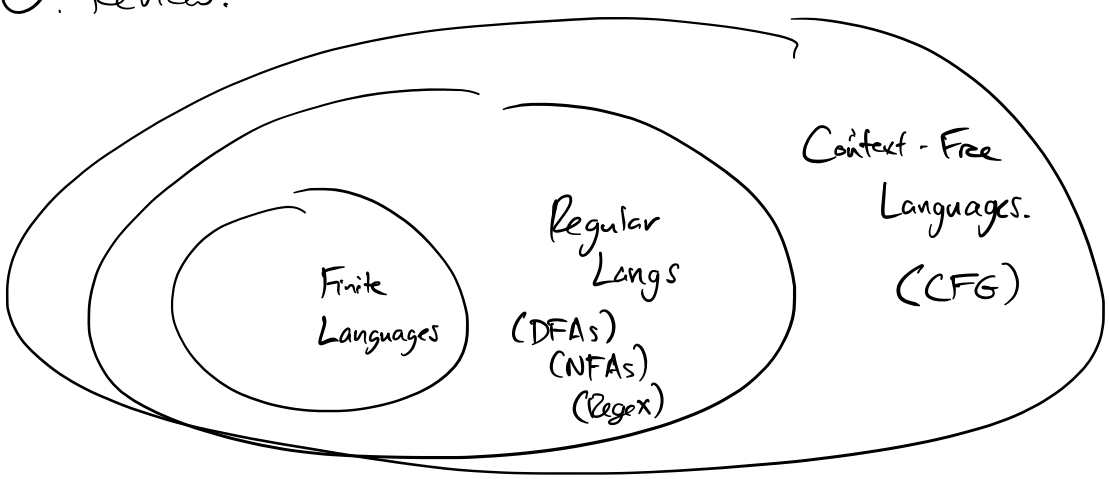
- HW #3 due Trees
- Virtual OH tonight @ 5:30 pm (+ HW1 review)

Today:

0. Review
1. Pushdown automata: automata w/ memory?
2. PDAs recognize the context-free languages (CFLs).
   - 2.1 CFG → Pushdown automaton
   - 2.2 PDA → CFG. (stated w/o proof.)

---

0. Review.



Grammar reminder:

$G = (V, \Sigma, R, S)$ — start

variables, terminals, rules

Procedure: exchange single variables for new strings of variables and terminals until we get a terminal string.

$V = \{S, A, B\}$

$\Sigma = \{1, 2, +, =\}$

$R = \{ S \to 1A2$

$A \to 1A1 \mid B$ (this is shorthand for "OR")

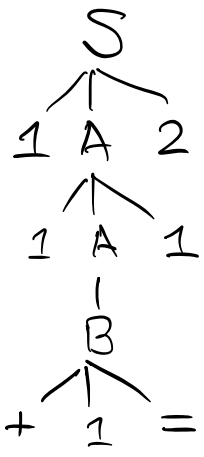
$B \to +1=$ $\}$

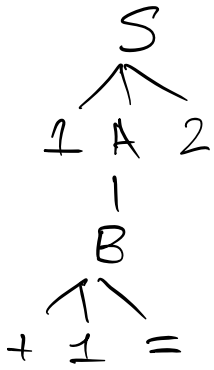
$A \to 1A1$

$A \to B$

$S \Rightarrow 1A2 \Rightarrow 11A12 \Rightarrow 11B12 \Rightarrow 11+1=12$

equivalently:





"How do I choose a rule"?

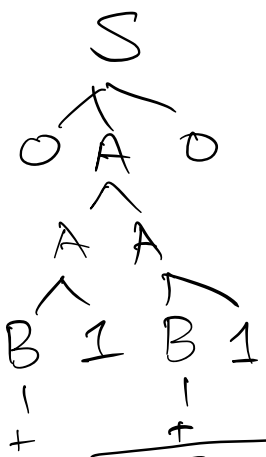
All choices are OK.

$S \to 0A0 \mid B1 \mid SS$

$A \to AA \mid 1B \mid B1$

$B \to + \mid \%$

0 + 1 + 1 0

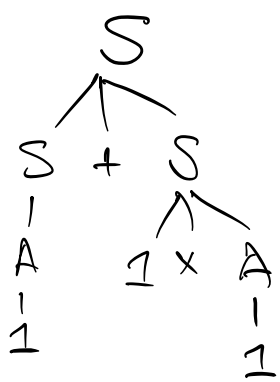


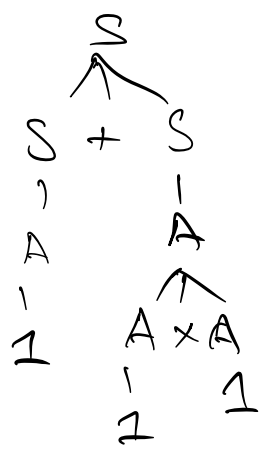
01+1+0 ✓

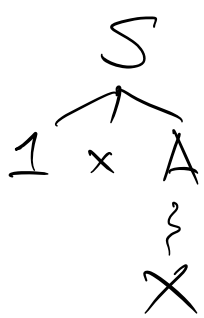
$S \to S+S \mid 1xA \mid A$

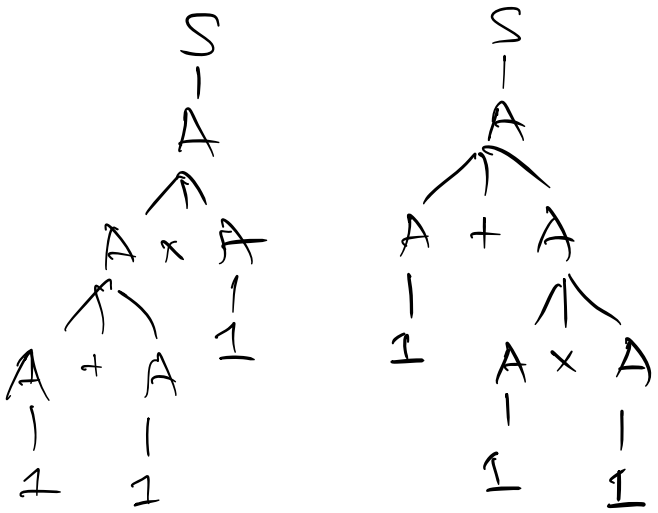
$A \to AxA \mid A+A \mid 1$

Derive 1+1x1?









1. Pushdown Automata: Automata w/ memory!

$$A = \{0^n 1^n \mid n \geq 0\}$$

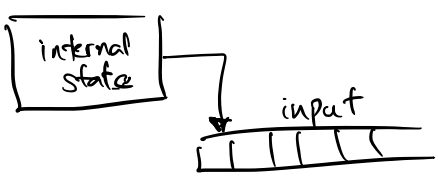
Program for recognizing strings in A:

```
balance = 0
while (next char 0):
    balance += 1
while (next char 1):
    balance -= 1
if balance == 0:
    YES
```

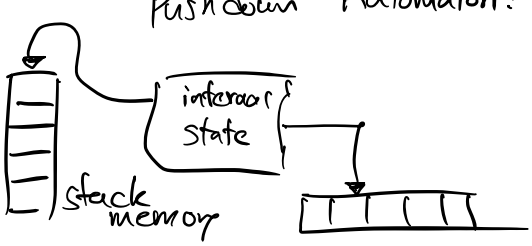
// DFA/NFA can't simulate this — we can't store 'balance' in any fixed finite # of states.

Solⁿ: Give our automaton access to a stack memory.

DFA/NFA:



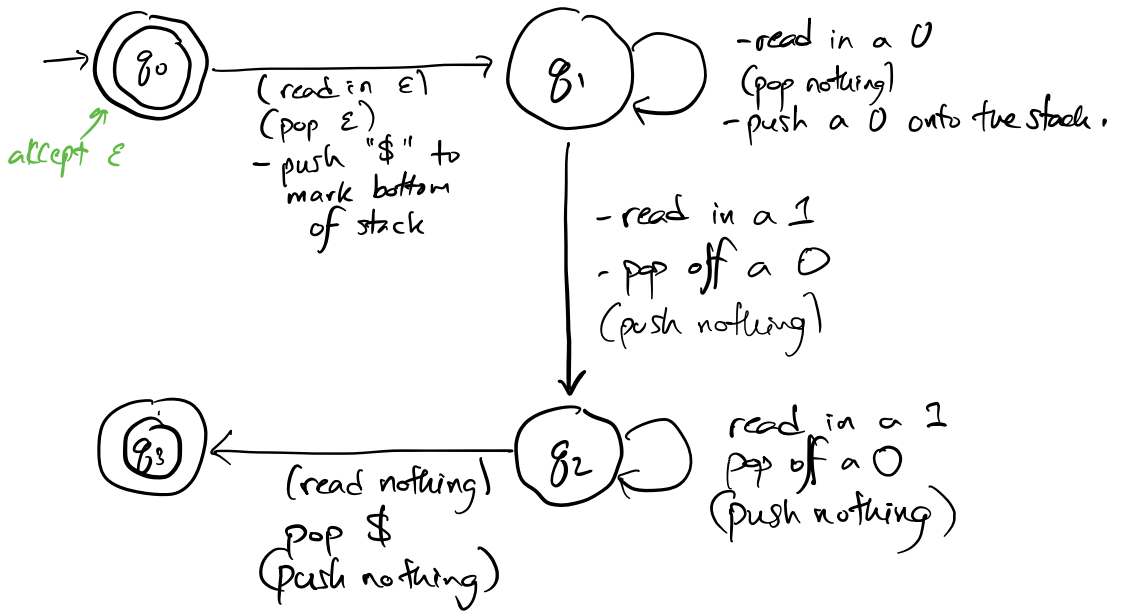
Pushdown Automaton:



Each step of computation: (PDA)

(1) read in an input char (or ε)
(2) pop something off the top of stack (or ε)
(3) move to a new state
(4) push something onto the stack.

Informal state diagram for $\{0^n 1^n \mid n \geq 0\}$:



(As before, we accept if at least one branch of computation is at the accept state after reading in all input.)

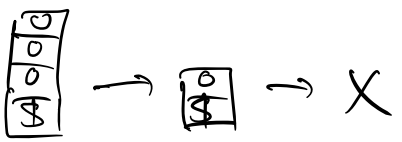
We can see only our current state, the next input symbol, and what pops off the stack.

Which branch?

On 000111, a stack trace:

push 

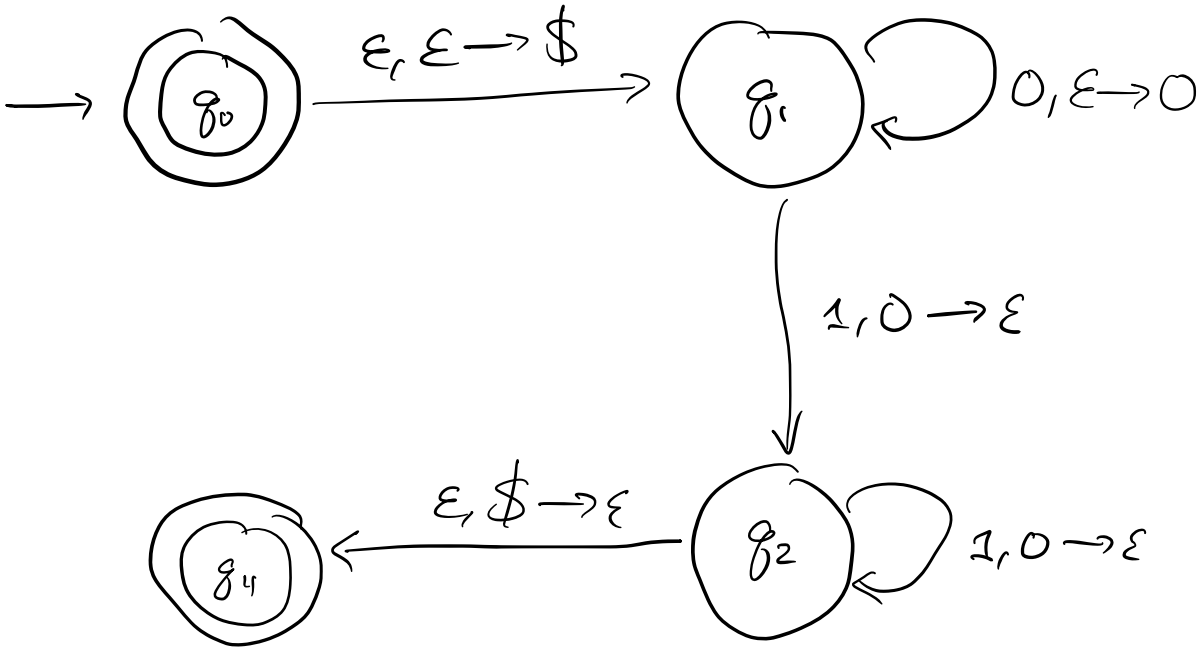
On 0011:



(why $ is important)

On 0011$\downarrow$1:

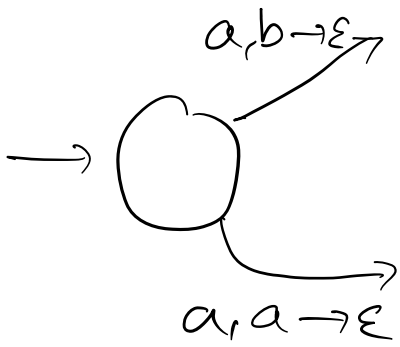
$\rightarrow$ X



a, b → c : "take this edge by
(1) reading in an a from input,
(2) popping a b off the stack,
(3) pushing a c onto the stack."

On string 11? Branch dies in q_1

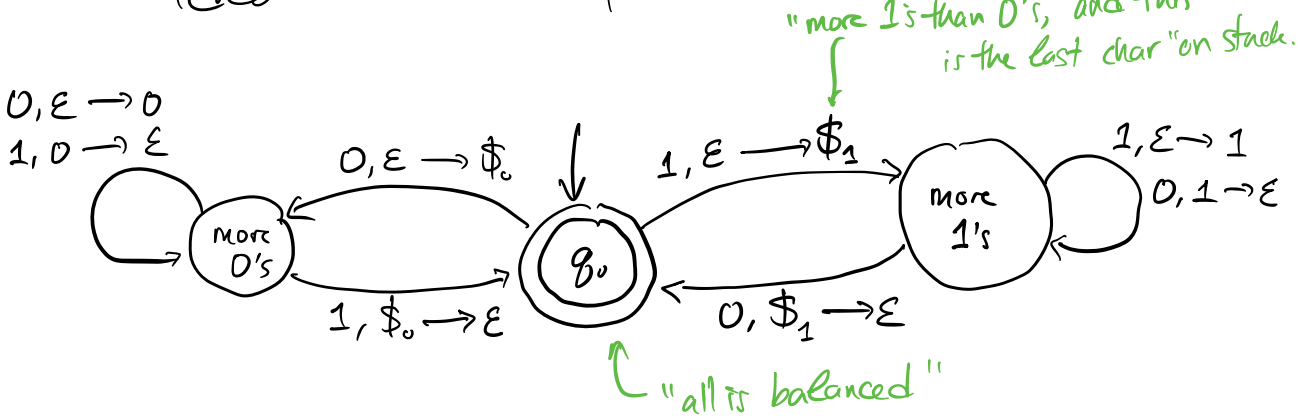
still have nondeterminism!



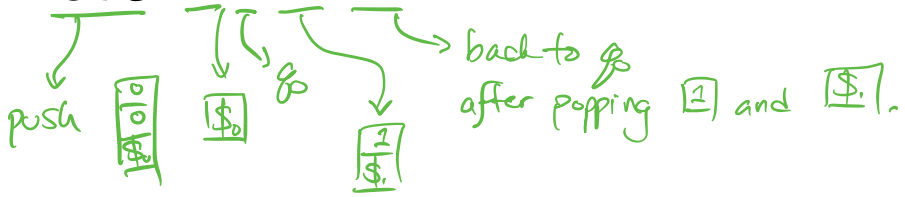
two branches, each with a stack.

Example. $B = \{ w \in \{0,1\}^* \mid$ w has the same number of 0's and 1's. $\}$

Idea: count "up" and "down"



Example: 000 1111100



Break: back at 2:40

PDA Puzzles.

- Follow $a, b \to c$ if we can read a and pop b
- accept if some branch accepts after reading in all input.

(Specify $\Sigma = \{0, 1\}$)

(1) PDA state diagram that pushes all input on stack, then accepts.

(Recall: $a, b \to c$ = "read a, pop b, push c"
$a, \varepsilon \to a$ = "read a, pop nothing, push a on stack")

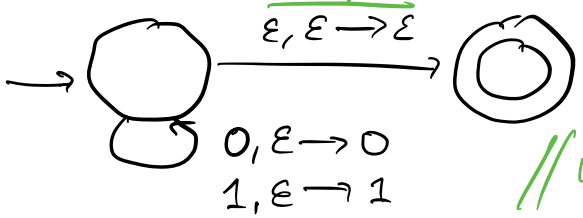
(2) PDA that pushes all input onto the stack,
pops all input off the stack,
then accepts.

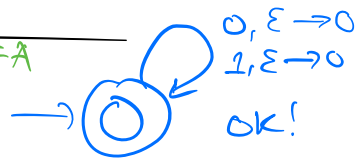
(3) PDA for $C = \{ww^R \mid w \in \{0,1\}^*\}$

(++) A PDA for $D = \{a^i b^j c^k \mid i = j \text{ OR } j = k,$ for $i, j, k \geq 0\}$,
on $\Sigma = \{a, b, c\}$.

---

(1)

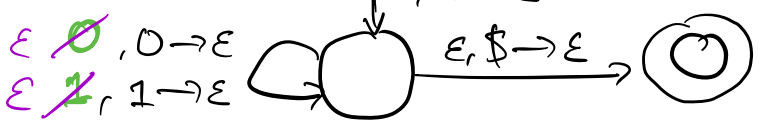


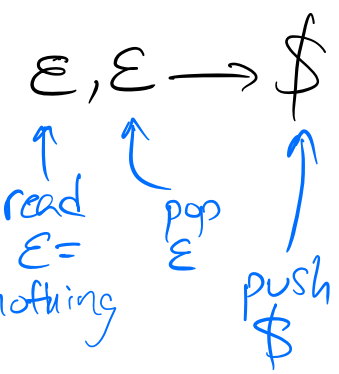


// why won't this accept before pushing all inputs?

(2)

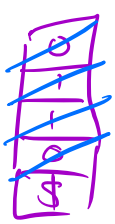
(3)





examples: 0110|0110

ε, ε → ε



011|001
$\varepsilon, \varepsilon \to \varepsilon$

On ε, stack:
$\$ \to \emptyset$

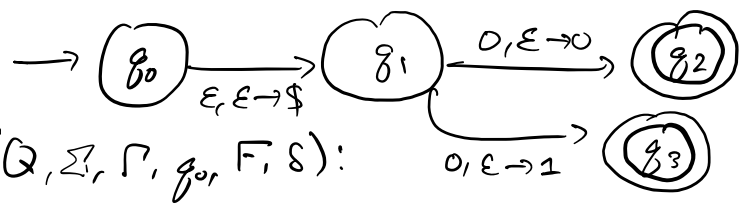
Def (PDA, Formal.) A Pushdown Automaton is a **6-tuple** $(Q, \Sigma, \Gamma, q_0, F, \delta)$, where:

Q is a set of states,
Σ is the input alphabet,
Γ is the stack alphabet,
q_0 is the start state,
F is the set of accept states
and $\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)^*$

(Σ_ε: $\Sigma \cup \{\varepsilon\}$; Γ_ε: $\Gamma \cup \{\varepsilon\}$; pair: new state, item to push)

"give me a state, an input symbol (or ε), and a stack symbol (or ε), and I'll give you a set of pairs (new state, thing to push)."

* what if we did $Q \times \Gamma_\varepsilon$? "DPDA" Sipser 2.4



PDA $P = (Q, \Sigma, \Gamma, q_0, F, \delta)$:

$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{0, 1\}$
$\Gamma = \{0, 1, \$\}$
$F = \{q_2, q_3\}$

$\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \$)\}$
$\delta(q_1, 0, \varepsilon) = \{(q_2, 0), (q_3, 1)\}$
all other inputs, δ maps to $\emptyset$.

Our PDA accepts an input $w = w_1 w_2 \cdots w_n$, with each $w_i \in \Sigma_\varepsilon$ if there is a sequence of states $r_0, r_1, \cdots r_n \in Q$,

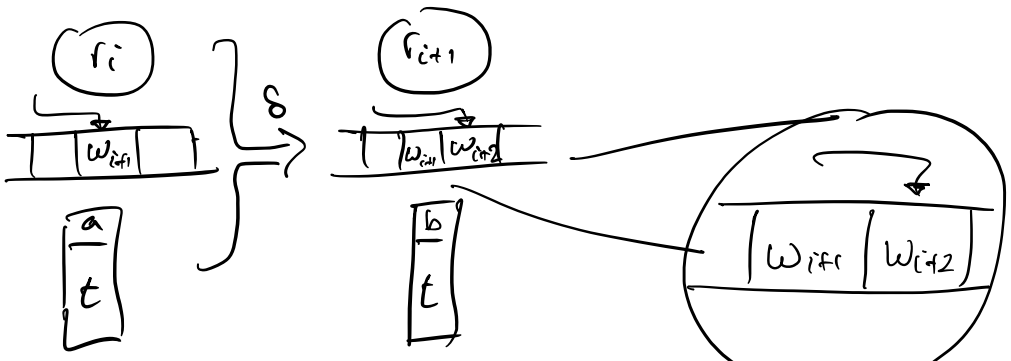
and a sequence of strings (stack states) $s_0, s_1, \cdots s_n \in \Gamma^*$ such that:

(1) $r_0 = q_0$, $s_0 = \varepsilon$, and $r_n \in F$,

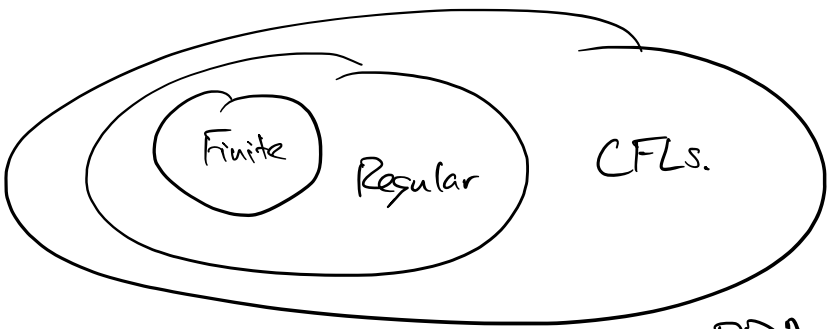
(2) for $i = 0, 1, 2, \cdots, n-1$

$$\delta(r_i, w_{i+1}, a) \ni (r_{i+1}, b)$$

where $s_i = at$ and $s_{i+1} = bt$ for $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$.



Break: Back at 3:33



PDAs?

Theorem: PDAs recognize exactly the CFLs.

Lemma 1. Any PDA $\Rightarrow$ Equivalent CFG. (Sipser pp. 121-124)

Lemma 2. Any CFG $\Rightarrow$ Equivalent PDA. (Sipser Lemma 2.21)

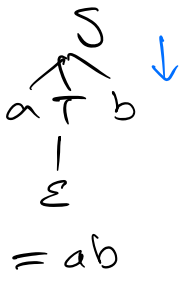
↳ sketch below.

Idea: CFG ⇒ PDA.

PDA will nondeterministically guess all derivations of the input string.

Some branch will accept ⟷ input string has a legitimate derivation in this CFG.

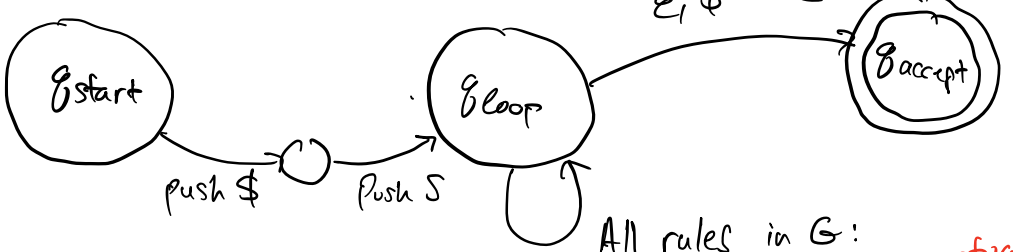
$$G: \quad S \to aTb \mid b$$
$$T \to Ta \mid \varepsilon$$



on input ~~aba~~:
- push $\boxed{\begin{matrix}S\\\$\end{matrix}}$
- one branch
- $a, a \to \varepsilon$

- one branch $T \to \varepsilon$

- $b, b \to \varepsilon$
- $\varepsilon, \$ \to \varepsilon$



All rules in G:

$\varepsilon, S \to aTb$

$\varepsilon, S \to b$

$\varepsilon, T \to Ta$

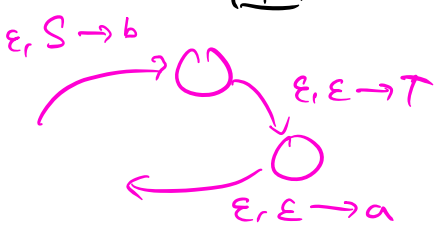
$\varepsilon, T \to \varepsilon$

Matching rules:

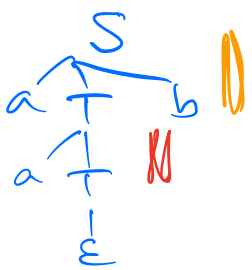
$a, a \to \varepsilon$

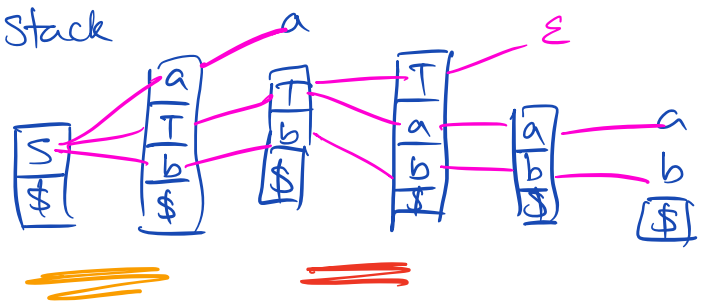
$b, b \to \varepsilon$

// actually 3 states, 3 transitions. (in reverse)



aab





Video: LR #6: Pushdown Automata.