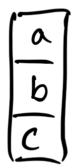
COMS W3261 — Lecture 6, Part 2/2:

Pushdown Automata.

Idea: a stack of symbols can be used as a simple memory.

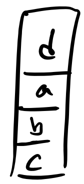
Stack. $a, b, c \in \Gamma$. || stack alphabet
capital gamma γ

access to
a only.



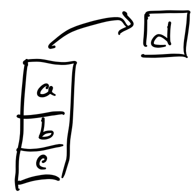
1. Can push
an element onto
the stack

push 'd'



2. Can pop
from top of
the stack.

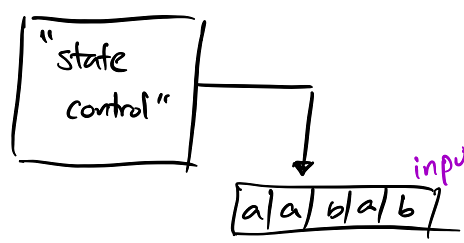
pop



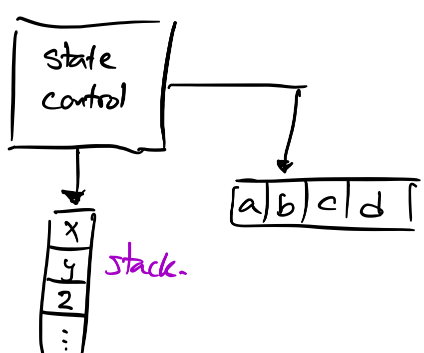
- Access only topmost element. (Pushdown automaton only knows items it pops.)
- Unlimited size.

Picture of a Pushdown Automaton (PDA) / comparison.

NFA/
DFA :



PDA :



On each computational step, our PDA (1) reads in an input symbol,

(2) reads/changes the current state, and (3) reads/writes from the stack.

- <u>(Nondeterministic) Pushdown Automaton state diagram</u>

Goal: Build a PDA for $\{0^n 1^n \mid n \geq 0\}$.

We will write each transition in our state diagram as

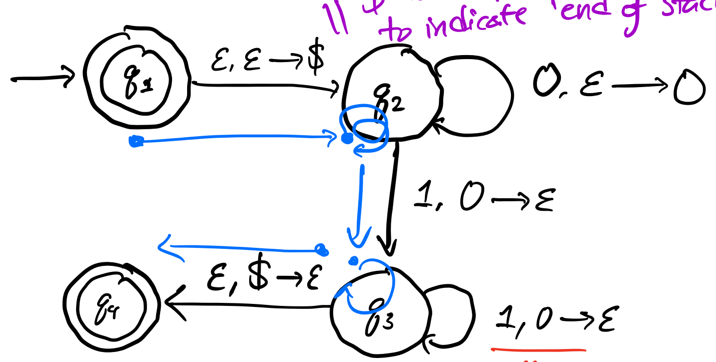
$$a,\ b \longrightarrow c$$

"read in the input symbol a" — "pop b from the top of the stack" — "push c onto the top of the stack"

We can use ε to indicate an ε-transition, or pushing/popping the empty string.

Idea: Push 0's onto the stack as we read them in. Then, we pop a 0 off the stack for every 1 we read in. Accept if and only if we read our last 1 when we pop our last zero.

// \$ is a special stack symbol we use to indicate 'end of stack'



// if we run out of 0's — can't take this edge.

Test strings.

00110

state	stack
q_1	ε
q_2	\$
q_2	0\$
q_2	00\$
q_3	0\$

state	stack	
q_3	\$	✓
q_4	ε	
$\emptyset$	ε	✗

Def. (Formal definition of Pushdown Automaton.)

(Idea: crucial part of def'n will be transition function. Let Σ_ε be Σ, the input alphabet, union $\{\varepsilon\}$, and introduce Γ_ε for $\Gamma \cup \{\varepsilon\}$, where Γ denotes the stack alphabet.

Reminder: $\mathcal{P}$ denotes the power set, the set of all subsets.

$\mathcal{P}(\{a,b,c\}) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\}\}$.)

A pushdown automaton is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$,

where Q is a finite set of states,

Σ is a finite input alphabet,

Γ is a finite stack alphabet,

q_0 is the start state,

$F \subseteq Q$ is the set of accept states,

and $\delta : \underbrace{Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon}_{} \longrightarrow \underbrace{\mathcal{P}(Q \times \Gamma_\varepsilon)}_{}$,

map from a state, an input symbol and a popped stack symbol

to any set of (go to this state, push this symbol) pairs.

Our PDA accepts an input string $w = w_1 w_2 \cdots w_n$, where each $w_i \in \Sigma_\varepsilon$, if there exists a sequence of states $r_0, r_1, \ldots, r_n$ and also strings $s_0, s_1, s_2 \cdots s_n \in \Gamma^*$ such that:

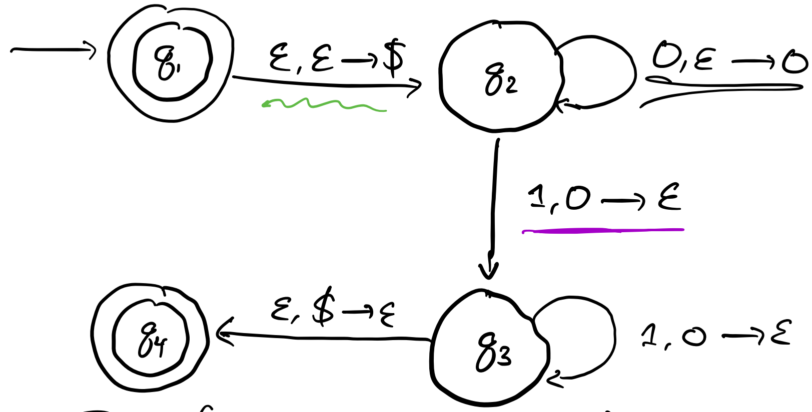
$$- \underline{r_0 = q_0}, \ \underline{r_n \in F}, \ \underline{s_0 = \varepsilon},$$

for $i = 0, 1, \ldots, n-1$ we have $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$,

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

Example: 6-tuple def'n of our previous state diagram.

(Recognizes $\{0^n 1^n \mid n \geq 0\}$)



Formally: $P = (Q, \Sigma, \Gamma, \delta, q_1, F)$, where:

$Q = \{q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, (1), \$\}$ optional // if designing a PDA - feel free to use any symbols in your stack alphabet.

$F = \{q_1, q_4\}$

(Recall: $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$)

δ is defined as follows:

$\delta(q_1, \varepsilon, \varepsilon) = \{(q_1, \$)\}$ // set of states

$\delta(q_2, 0, \varepsilon) = \{(q_2, 0)\}$

$\delta(q_2, 1, 0) = \{(q_3, \varepsilon)\}$

$\delta(q_3, 1, 0) = \{(q_3, \varepsilon)\}$

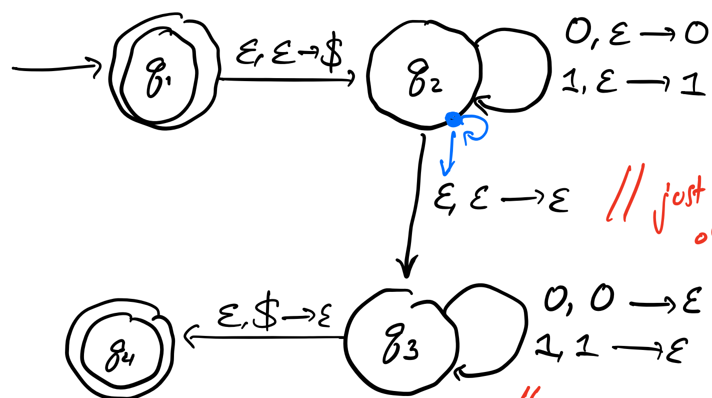
$\delta(q_3, \varepsilon, \$) = \{(q_4, \$)\}$

$\delta(\cdot, \cdot, \cdot) = \emptyset$ for all other inputs.

<u>Example</u>. Building a PDA that recognizes $\{ww^R \mid w \in \{0,1\}^*\}$. ($w^R$ indicates w reversed.)

<u>Idea:</u> Similar to $\{0^n 1^n \mid n \geq 0\}$. Push symbols onto the stack, then

nondeterministically guess the midpoint of the input, then accept if the symbols we pop exactly match the remaining input.



// just like our old ε-edge.

// if in q_3, read '0' on input and have '1' on stack — die.

0110.

state	stack
q_1	ε
q_2	\$
q_2	0\$
q_2	10\$
q_3	10\$
q_3	0\$
q_3	\$
q_4	ε

✓

// ignoring branches that take $\varepsilon, \varepsilon \to \varepsilon$ transition at the wrong time

---

Next time: prove PDAs are equivalent in power to CFGs.
PDAs recognize CFLs.

Reading: Sipser 2.2.