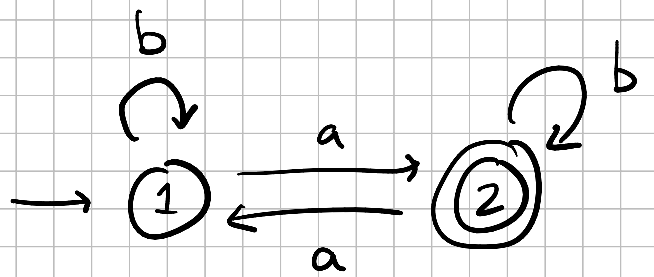
CS321 → DFAs



Start in state 1

read input chars, 1 at a time, L → R

for each char, follow corresponding arrow

at end of input if

◎ → output yes

○ → output no

ex: input = abba

$1 \xrightarrow{a} 2 \xrightarrow{b} 2 \xrightarrow{b} 2 \xrightarrow{a} 1$ = NO

this machine says yes ⟺ input has odd # of a's

State 1 means "seen even # a's so far"

State 2 " odd "

Design machine that says yes ⟺ input contains "aa" as substring

test cases:

bbb → no

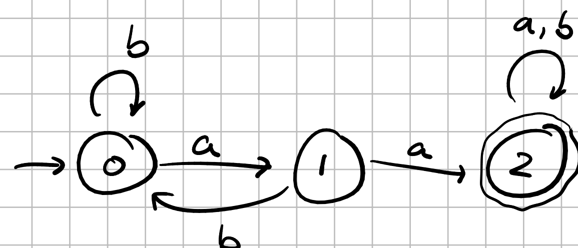
aba → no

baaaa → yes

a → no

ababab → no

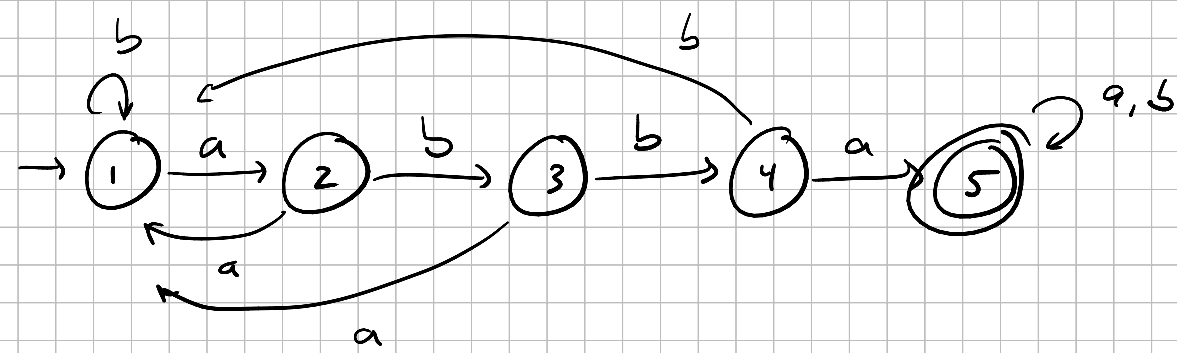
abaaba → yes



state i means "seen i a's in a row" (kind of)

Design machine that says yes $\iff$ input contains "abba" as substring

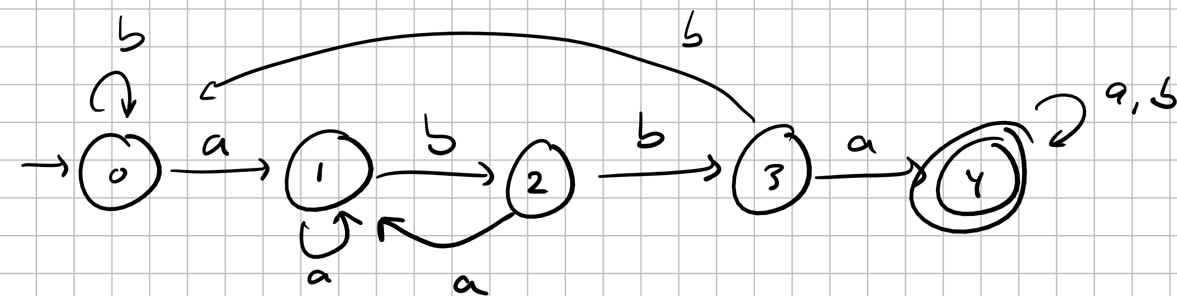
"obvious" answer:



Problem: aabba $\quad 1 \xrightarrow{a} 2 \xrightarrow{a} 1 \xrightarrow{b} 1 \xrightarrow{b} 1 \xrightarrow{a} 2 = NO$

ababba $\quad$ similar problem

Fixed



States 0 through 3:

state # i means: last i chars of input

= first i chars of abba

Def: an alphabet is finite set of characters

ex: $\{0,1\}$, $\{a,b\}$, $\{\text{ascii characters}\}$

Def: a string is an ordered seq. of characters

the empty string has zero characters, written ε

(not $\in$)

(some references use λ)

Def: concatenation is written as "multiplication"

ex: $x = abba$
$y = baa$ then $xy = abbabaa$

ex: $x\varepsilon = x = \varepsilon x$

1 automaton
2 automata

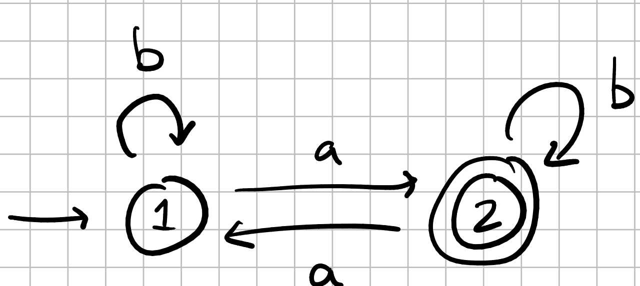
Def: A <u>Deterministic</u> <u>Finite</u> <u>Automaton</u> (DFA) is a tuple $M = (Q, \Sigma, \delta, s, F)$ where

- Q: finite set of states
- Σ = alphabet of input chars.
- δ = transition function $\delta : Q \times \Sigma \to Q$

  (takes (state, character) pair and outputs a state)

  "in state q, read char c $\longrightarrow$ go to state $\delta(q,c)$"
- s: start state ($\in Q$)
- F: accept states ($F \subseteq Q$)

Example



$Q = \{1, 2\}$

$\Sigma = \{a, b\}$

δ

1	a	2
1	b	1
2	a	1
2	b	2

$s = 1$

$F = \{2\}$