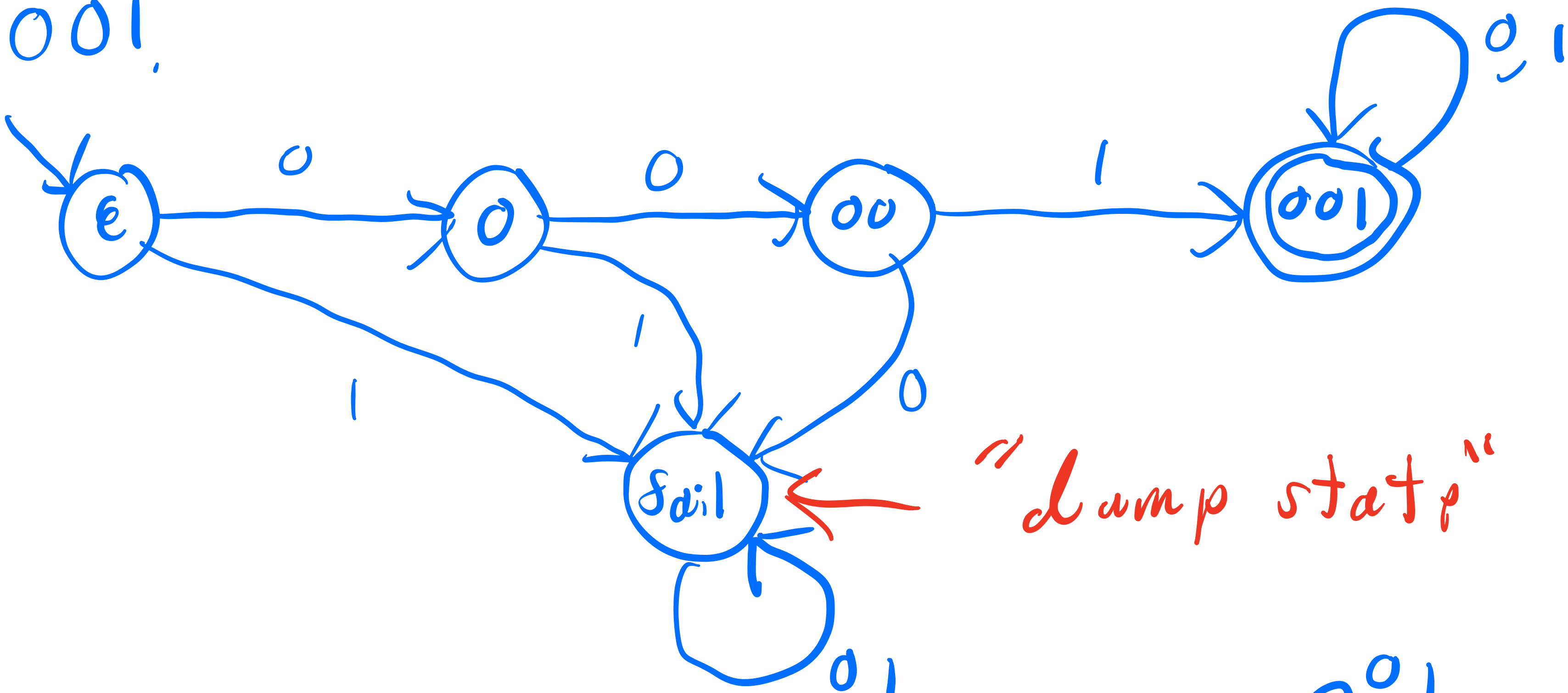
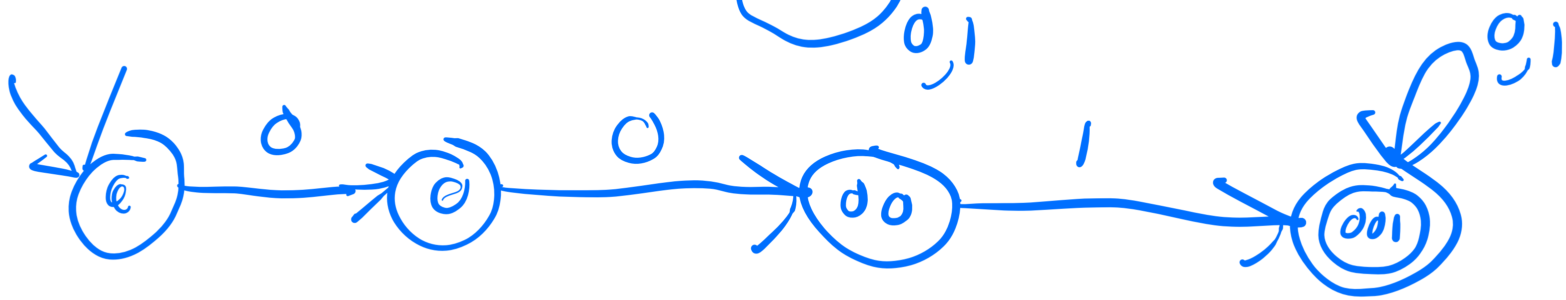
Mon Feb 2nd: 10th day registration deadline

Regular expression writing tips on website.

DFA Ex: All binary strings beginning with 001.

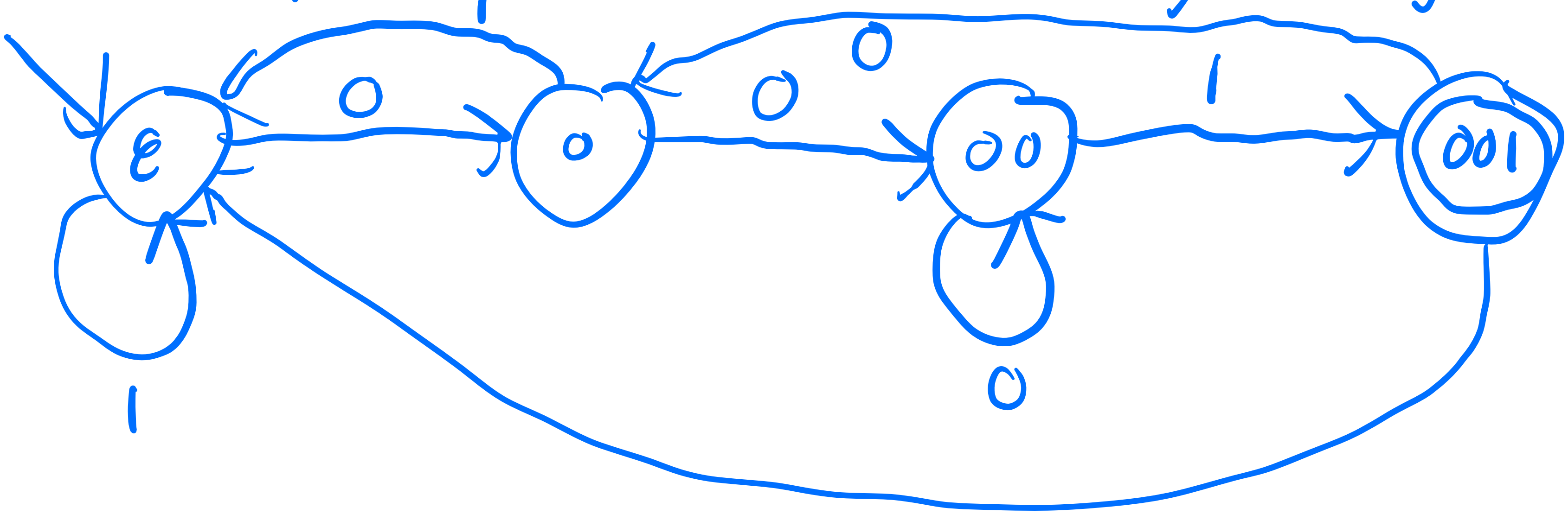




No transition ⇒ go to dump state, but you have to tell us you're doing that!

Ex: All binary string that end with 001.

states: what we're currently ending with.



A DFA over a fixed alphabet Σ can be described as $M = (Q, \delta, s, A)$.

- Q: finite set of states
- $\delta : Q \times \Sigma \to Q$ the transition function
- $s \in Q$: start state
- $A \subseteq Q$: accept states

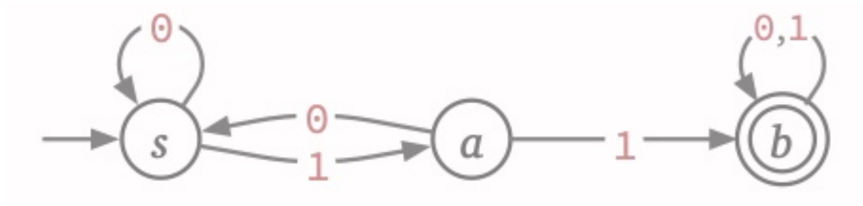
extended transition function $\delta^* : Q \times \Sigma^*$

$$\delta^*(q, w) := \begin{cases} q & \text{if } w = \varepsilon \\ \delta^*(\delta(q, a), x) & \text{if } w = ax \end{cases}$$

The language of/accepted by DFA M is

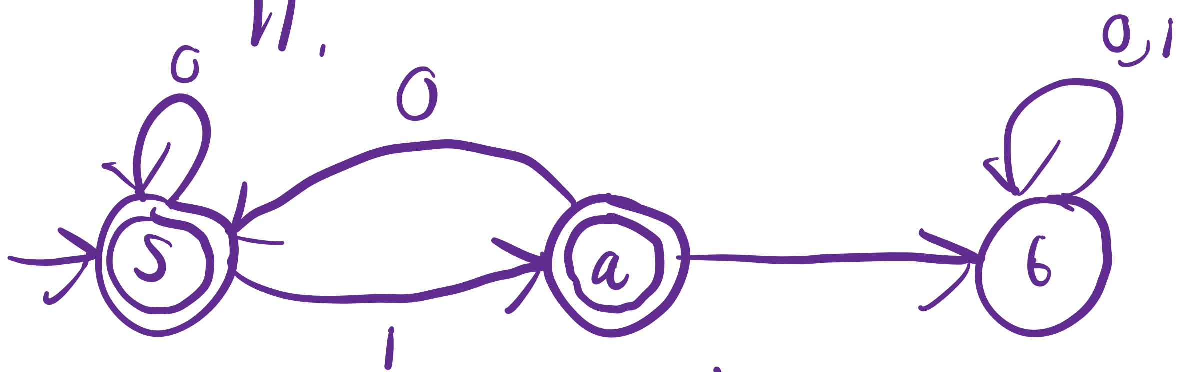
$$L(M) := \{ w \mid M \text{ accepts } w \}$$
$$= \{ w \mid \delta^*(s, w) \in A \}$$

Ex: M_{11} :



$L(M_{11})$: binary strings with substring 11.

$\overline{M}_{11}$:

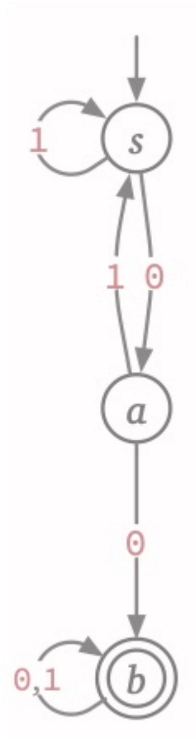


$$L(\overline{M}_{11}) = \overline{L(M_{11})} = \Sigma^* \setminus L(M_{11})$$

Given machine $M = (Q, \delta, s, A)$, there is a machine $\overline{M} = (Q, \delta, s, \overline{A})$ with $\overline{A} = Q \setminus A$

s.t. $L(\overline{M}) = \overline{L(M)}$.

M_{00}:



$L(M_{00})$: have 00 as a substring

Can we accept strings with both 11 & 00?

product construction

use states (p,q) where p from M_{00} & q from M_{11}

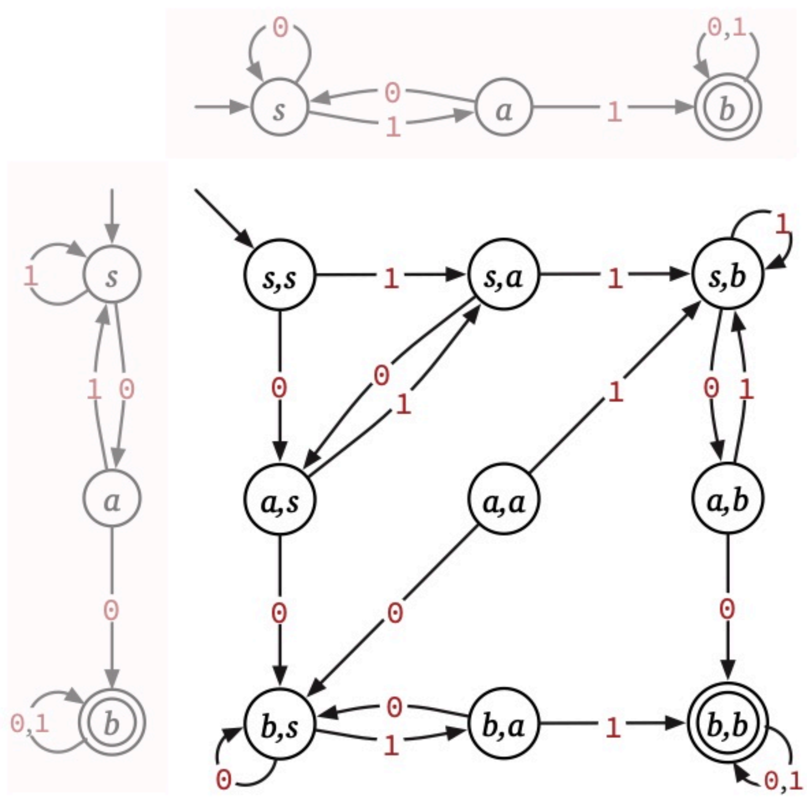
start from (s, s')

from M_{10} from M_{11}

transitions $(p,q) \xrightarrow{a} (p',q')$

if M_{00} has $p \xrightarrow{a} p'$ & M_{11} has $q \xrightarrow{a} q'$

accept (p,q) where M_{00} accepts p & M_{11} accepts q



Building a DFA for the language of strings containing both 00 and 11.

Given two machines $M_1 = (Q_1, \delta_1, s_1, A_1)$
$+ M_2 = (Q_2, \delta_2, s_2, A_2)$,

a <u>product</u> or <u>product construction</u>
is a machine $M = (Q, \delta, s, A)$ s.t.

- $Q = Q_1 \times Q_2 = \{(p, q) \mid p \in Q_1 + q \in Q_2\}$
- $\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$
- $s = (s_1, s_2)$

Lemma: $\delta^*((p, q), w) = (\delta_1^*(p, w), \delta_2^*(q, w))$

Induction!

Different A give different languages!

Want $L(M) = L(M_1) \cap L(M_2)$:

$$A = \{(p,q) \mid p \in A_1 \ \underline{\text{and}} \ q \in A_2\}$$

... $L(M_1) \cup L(M_2)$

$$A = \{(p,q) \mid p \in A_1 \ \underline{\text{or}} \ q \in A_2\}$$

inclusive

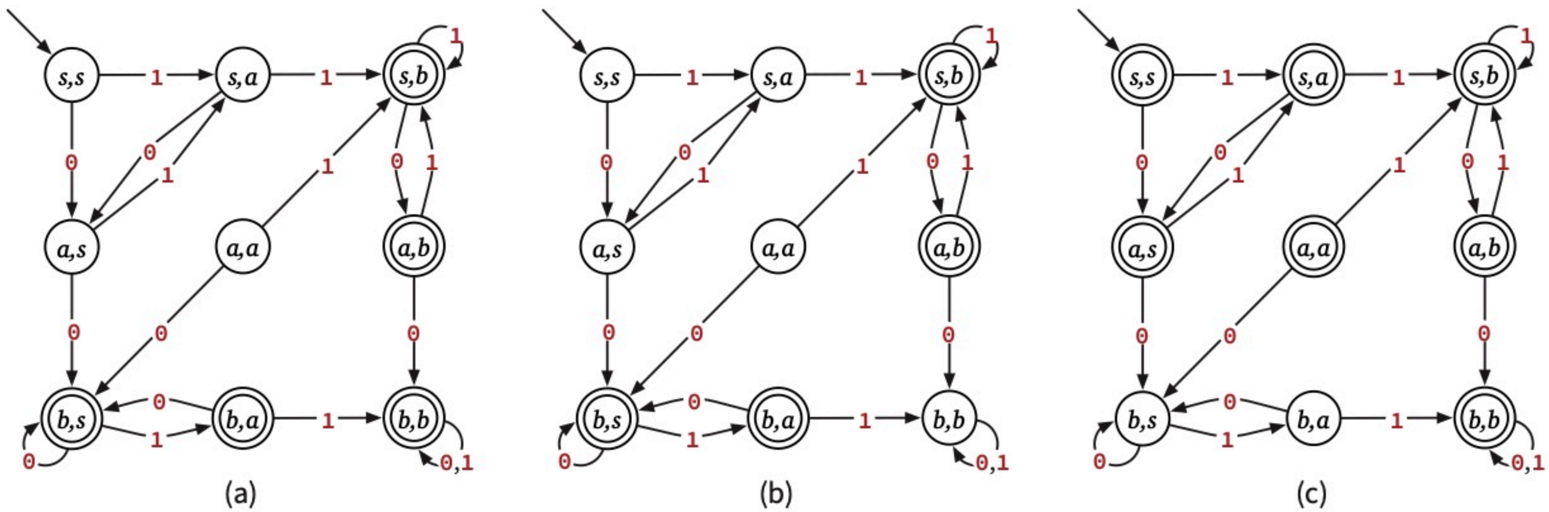
$L(M_1) \setminus L(M_2)$

$$A = \{(p,q) \mid p \in A_1 \ \underline{\text{and}} \ q \ \underline{\notin} \ A_2\}$$

symmetric diff.

$L(M_1) \oplus L(M_2)$

$$A = \{(p,q) \mid p \in A_1 \ \underline{\text{xor}} \ q \in A_2\}$$



DFAs for (a) strings that contain 00 or 11, (b) strings that contain either 00 or 11 but not both, and (c) strings that contain 11 if they contain 00. These DFAs are identical except for their choices of accepting states.

When you do a product, you <u>must</u> describe the accept states.

A language is "~~automatic~~ regular" if it is $L(M)$ for some DFA M.

Thm: Let L_1 + L_2 be ~~automatic~~ regular languages, then the following are ~~automatic~~ regular.

$$\overline{L_1} = \Sigma^* \setminus L_1$$

$$L_1 \cup L_2$$

$$L_1 \cap L_2$$

$$L_1 \setminus L_2$$

$$L_1 \oplus L_2$$

Thm (Kleene): The automatic language are precisely the regular languages.

Cor: Given two automatic languages L_1 & L_2,

$L_1 \cdot L_2$ & $(L_1)^*$ are automatic.

To prove L is regular:

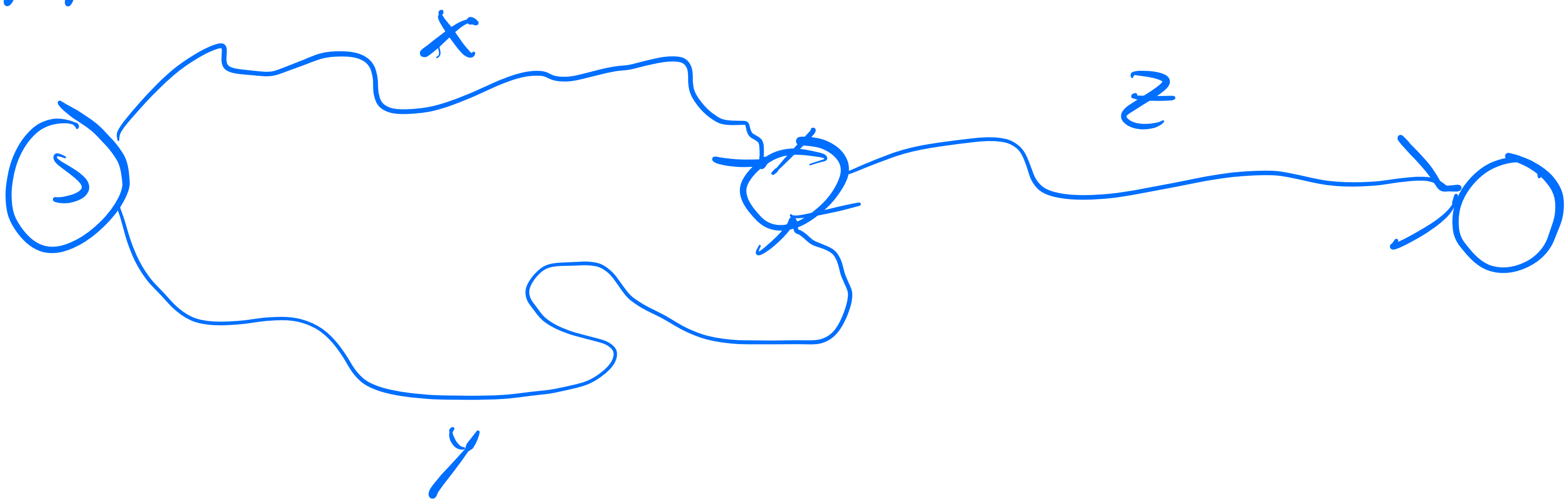
find a regular expression

or build a DFA

⋮

Suppose we have two string x & y & a DFA $M = (Q, \delta, s, A)$.

Suppose $\delta^*(s, x) = \delta^*(s, y)$.

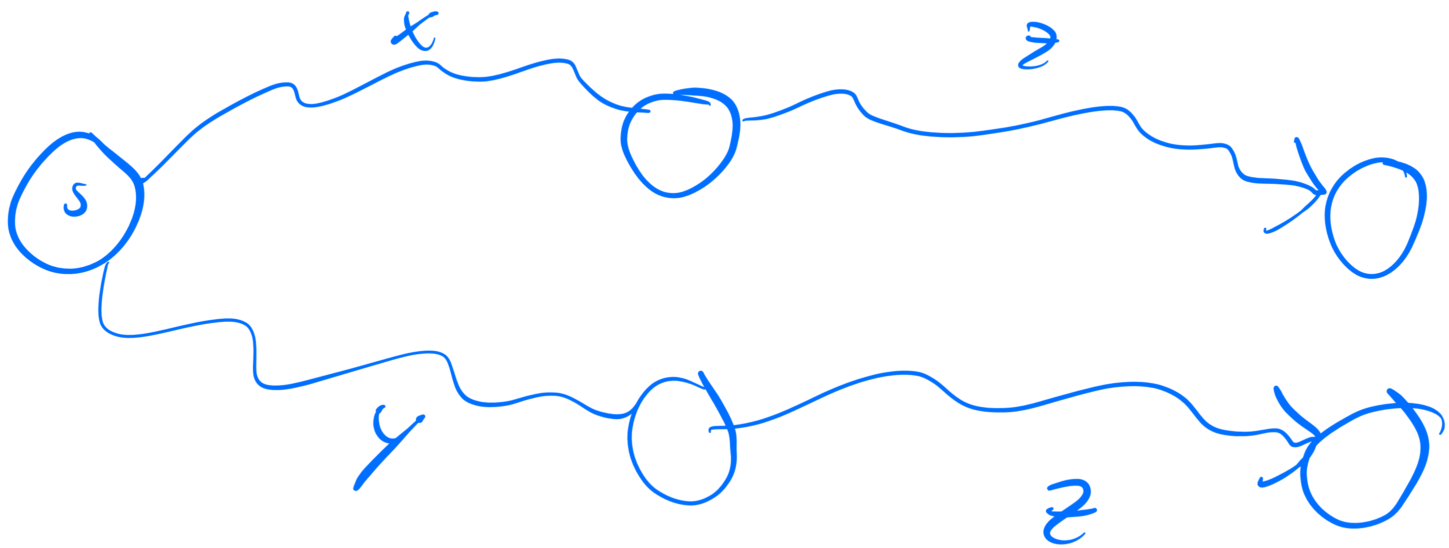


Let z be another string.

$$\delta^*(s, xz) = \delta^*(s, yz)$$

$\Leftarrow$ If $\delta^*(s, xz) \neq \delta^*(s, yz)$,

then $\delta^*(s, x) \neq \delta^*(s, y)$



Meaning M has ≥ 2 states.

Also, for any language L s.t.

$xz \in L$

$yz \notin L$, any DFA accepting L has ≥ 2 states.

Suppose we have a set F of strings & for any distinct x, y in F, there is a z s.t. $xz \in L$, ← some language
$yz \notin L$.

$\Rightarrow$ any DFA for L has $\geq |F|$ states.