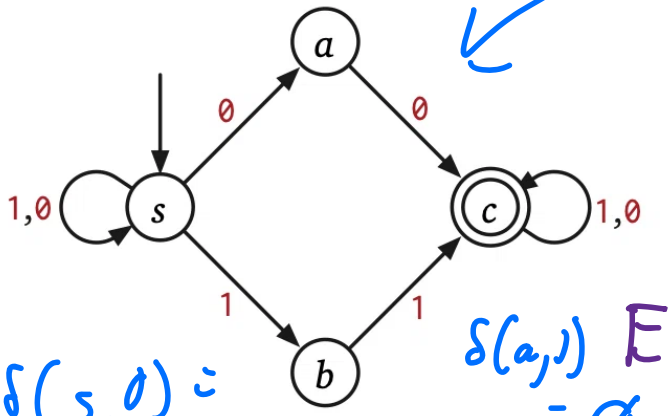
GPS 2 due today. Sep 8.
HW 2 due tomorrow, Sep 9.

any string with 00 or 11.

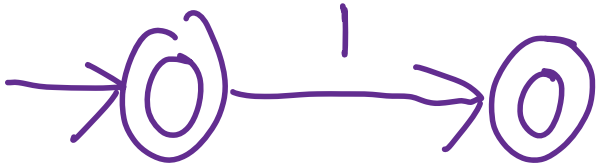


nondeterministic finite-state automata (NFAs)

$\delta(s,0) = \{s,a\}$

$\delta(a,1) = \emptyset$

Ex: 1010110 is accepted.



Accepts $\{\varepsilon, 1\}$

An NFA $M = (\Sigma, Q, \delta, s, A)$:

- input alphabet Σ
- finite set of states Q
- start state $s \in Q$
- accept states $A \subseteq Q$
- transition function $\delta : Q \times \Sigma \to \mathcal{P}(Q)$

  $(\text{i.e. } 2^Q)$

  (subsets of Q)

defines extended transition function $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$

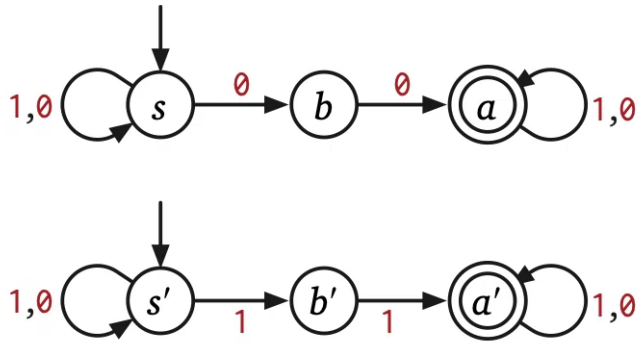
$$\delta^*(q, w) := \begin{cases} \{q\} & \text{if } w = \varepsilon \\ \bigcup_{r \in \delta(q,a)} \delta^*(r, x) & \text{if } w = ax \end{cases}$$

$L(M) := \{ w \mid \delta^*(s, w) \cap A \neq \emptyset \}$

i.e. accept if we <u>can</u> reach <u>any</u> accept state

Extensions:

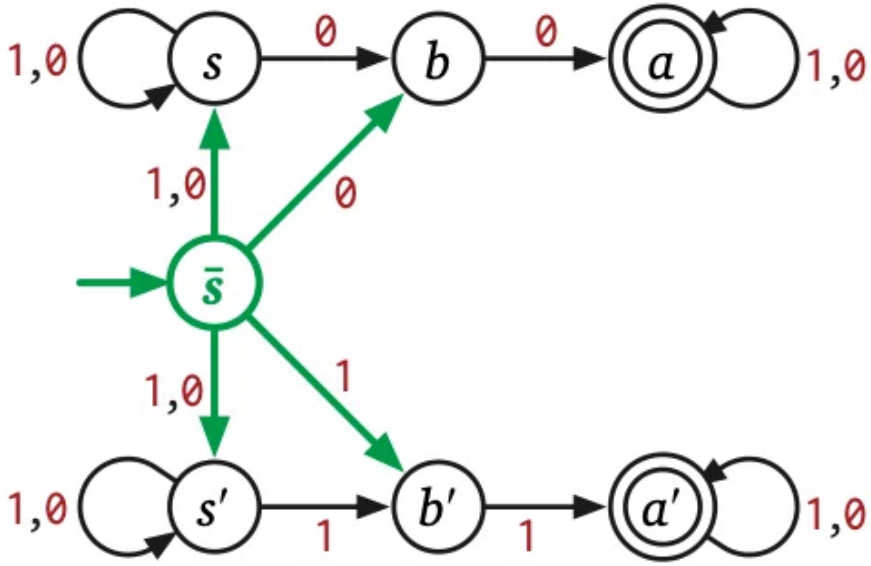
Multiple start states:



← one NFA with two start starts

Guess a start state & see if input string brings you to accept,

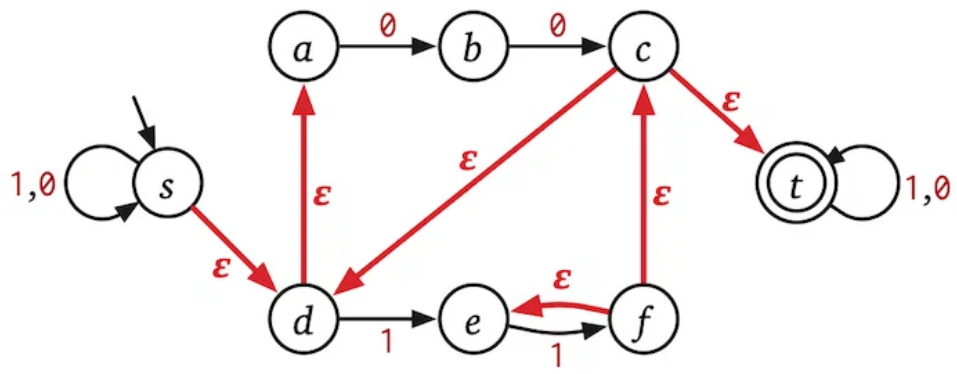
Formally: With start states $S \subseteq Q$, accept w iff $(\bigcup_{s \in S} \delta^*(s, w)) \cap A \neq \emptyset$.



Turn an NFA with multiple start states S to a regular NFA.

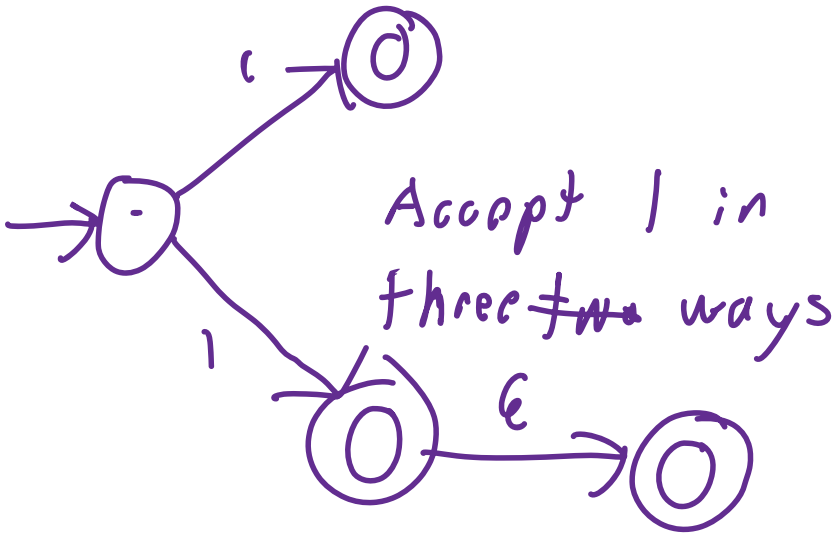
- add new state $\bar{s}$
- copy all transitions out of S to leave $\bar{s}$
- make $\bar{s}$ the one start state
- add $\bar{s}$ to A if $S \cap A \neq \emptyset$

ε-transitions:



You get the option of following an ε-transition without consuming a symbol.

Ex: 0(01|10



Accept 1 in three ways

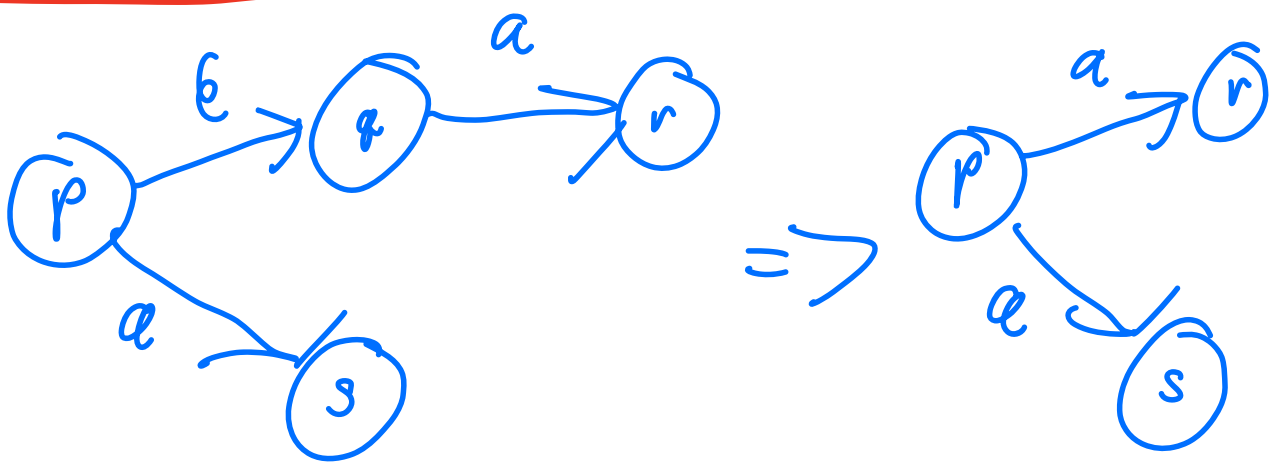
Formally: $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$

ε-reach(q): all states including q reachable from q using only ε-transitions

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

Can turn NFA $M = (Q, \delta, s, A)$ with ϵ-transitions into a regular NFA $M' = (Q, \delta', s, A')$

- $\delta'(p,a) = \bigcup_{q \in \epsilon\text{-reach}(p)} \delta(q,a)$
- $A' = \{p \mid \epsilon\text{-reach}(p) \cap A \neq \emptyset\}$ ← fixed after lecture ended!



Say your NFA has multiple start states and/or ϵ-transitions if you use

them.

Kleene's theorem:

"automatic" ≡ languages accepted by NFAs ≡ regular

if L is accepted by a DFA, it is accepted by an NFA.

the set of transitions has the one option.

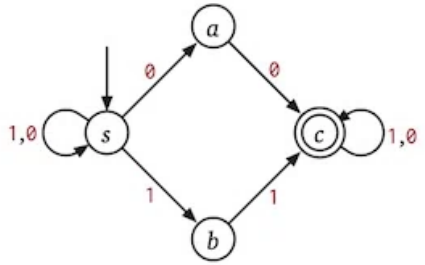
Given NFA $M = (Q, \delta, s, A)$, build DFA $M' = (Q', \delta', s', A')$

$$Q' = \mathcal{P}(Q)$$

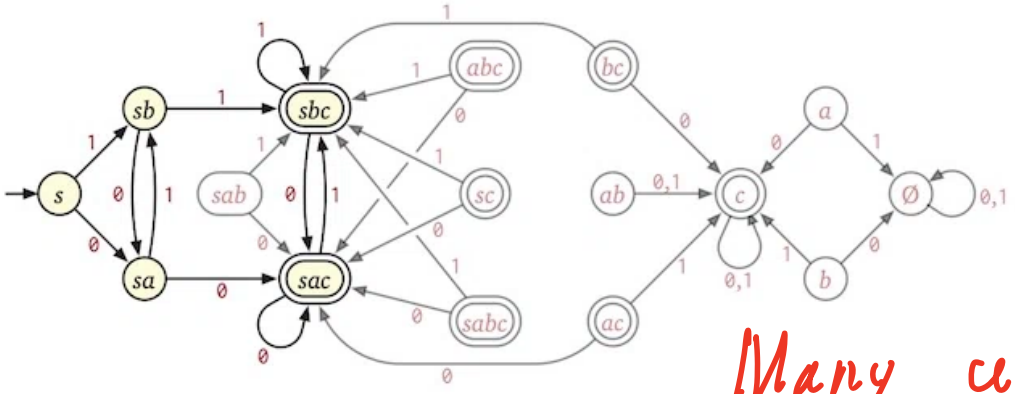
$$\delta'(P, a) = \bigcup_{q \in P} \delta(q, a)$$

$$s' = \{s\}$$

$$A' = \{ P \subseteq Q \mid P \cap A \neq \emptyset \}$$



NFA



DFA

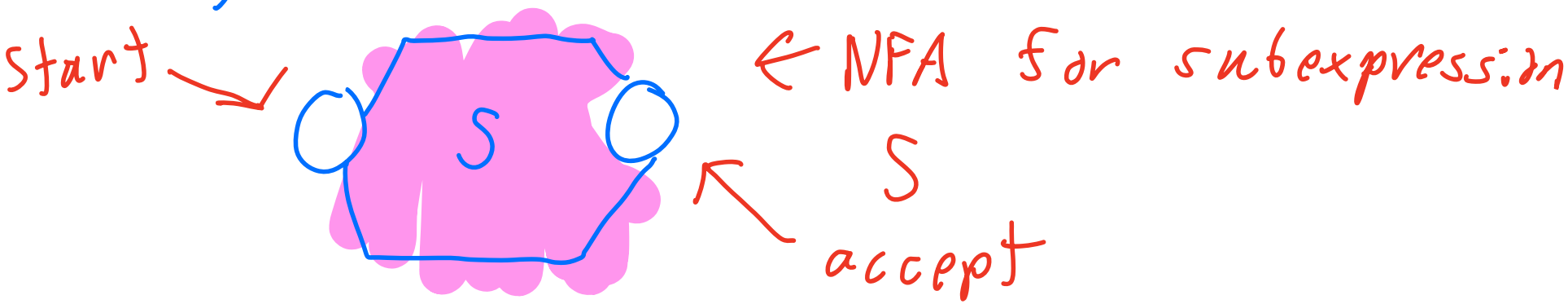
Many unreachable

DFA states!

All regular languages are accepted by NFAs (with ε-transitions)

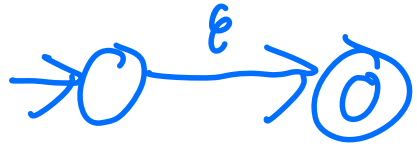
Thompson's algorithm:

Given a regular expression R, will build an NFA for its language with one accept state (& one start state) & they aren't the same state.

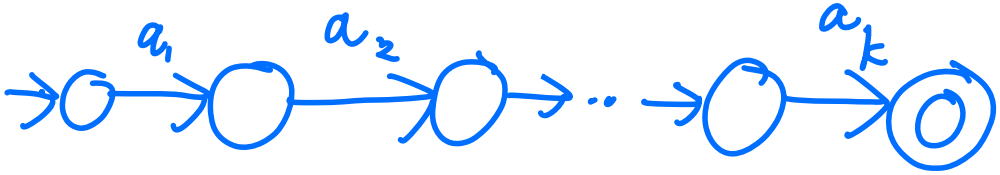


If $R = \emptyset$:

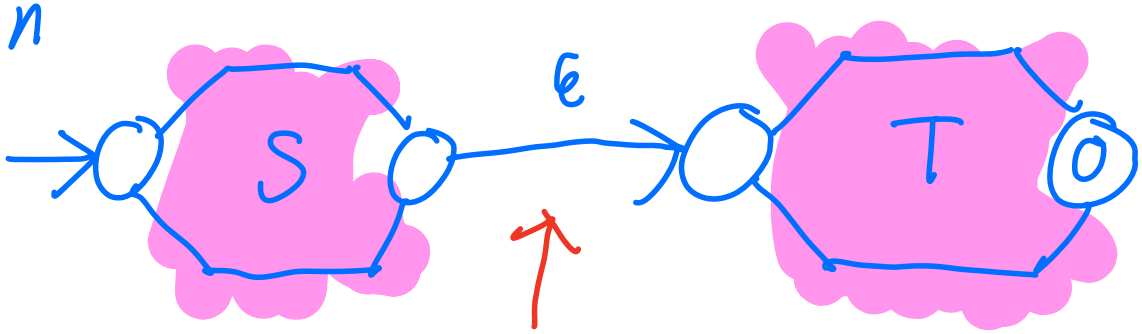
If $R = \epsilon$:



If $R = w = a_1 a_2 \ldots a_k$:

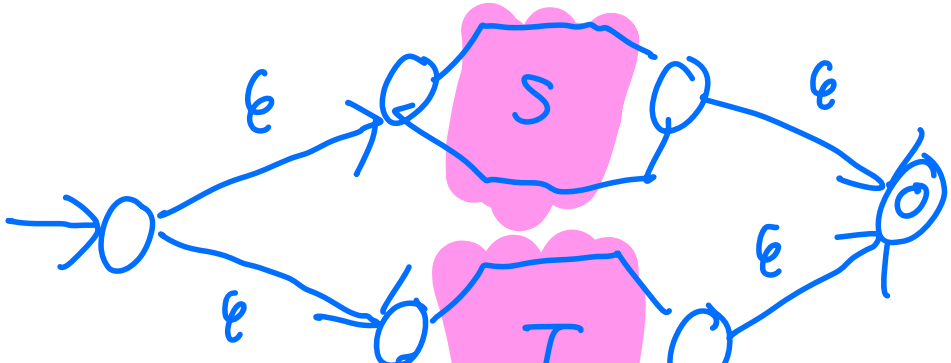


If $R = ST$: Build machines for S & T, & then

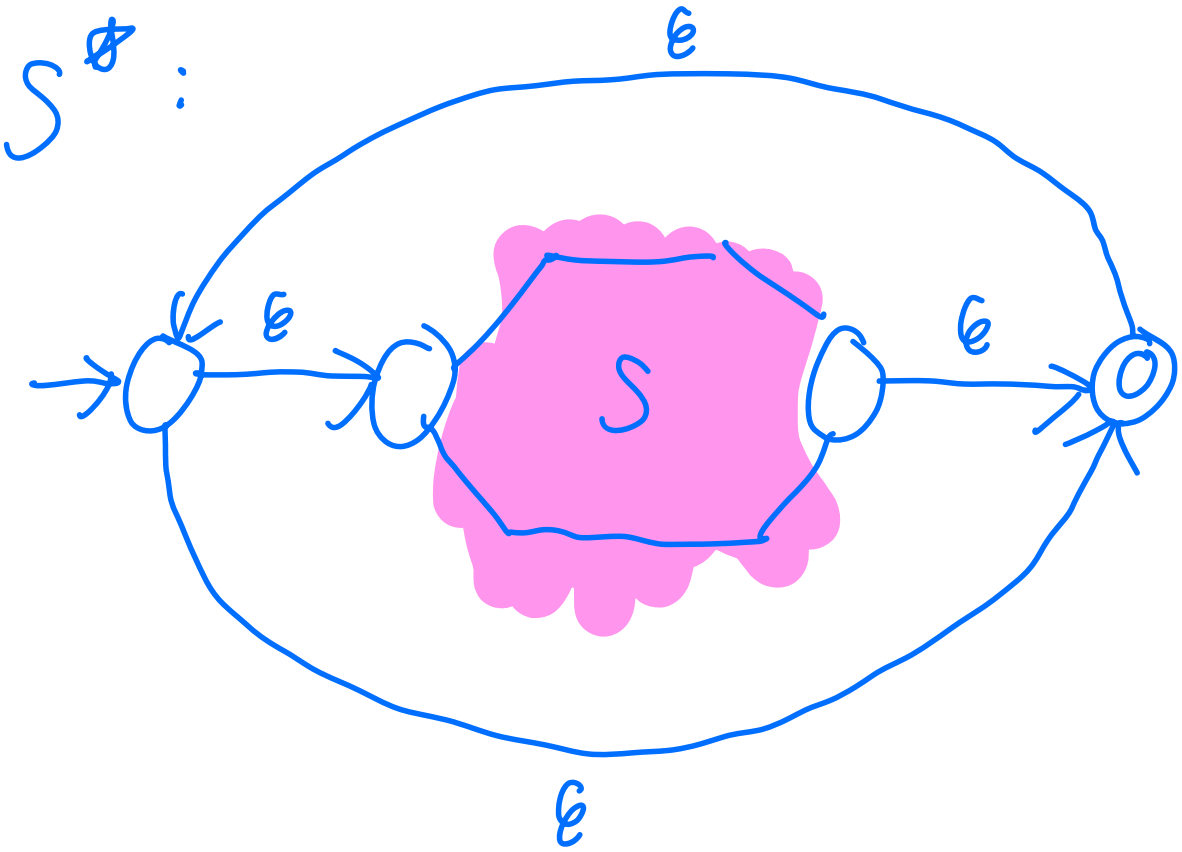


guess when to move on & then commit

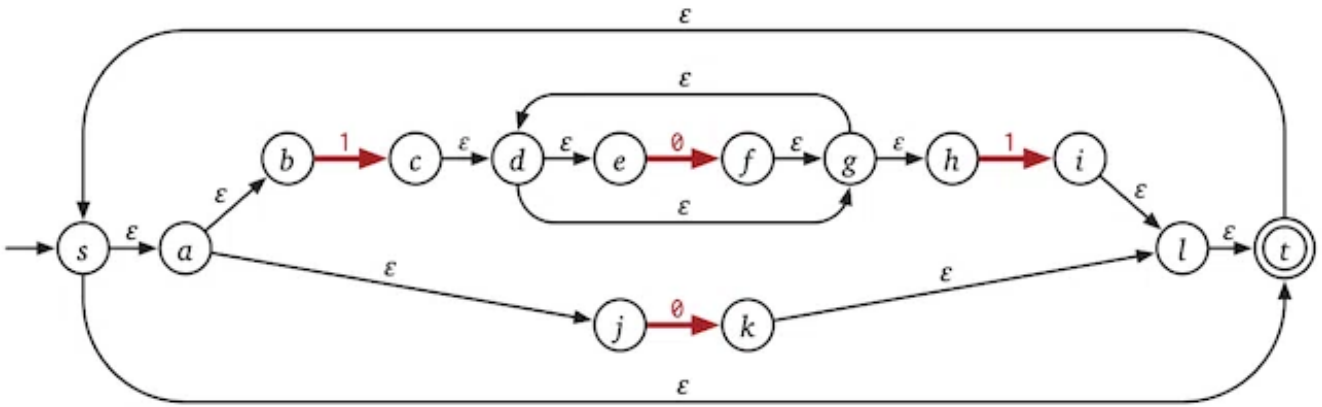
If $R = S + T$:



If $R = S^*$:



Ex: $(10^*1 + 0)^*$:



NFA => Regular expression...

Thursday!