Lecture 4

Thursday, 4 February, 2021 10:48

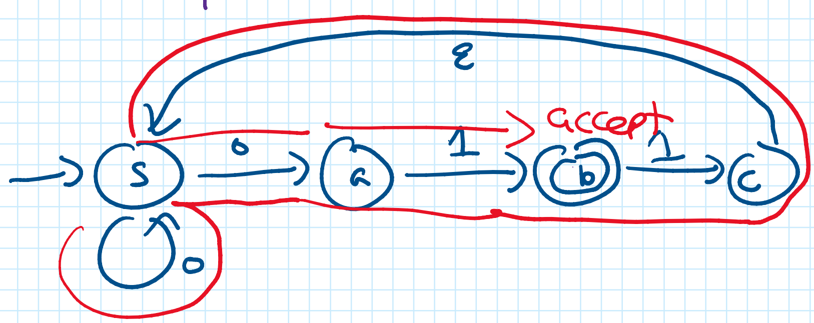
Non deterministic Finite Automata (NFA)

Recall that for DFAs:

- for each state, exactly one outgoing (transition) edge for each $a \in \Sigma$
- formalized by type sig $\delta : Q \times \Sigma \to Q$
  - $\to$ extended transition fn $\delta^* : Q \times \Sigma^* \to Q$
  - $\to$ $\delta^*(s,w)$ is exactly one state
    check if $\delta^*(s,w) \in A$.

In NFA: relax condition on transitions.
allow <u>any number</u> of transitions per $a \in \Sigma$.
also allow special ε-transitions
that can be taken for free without reading input.

ex.



001101

type sig of NFA transition: $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to P(Q)$

$\delta(s,0) = \{s,a\}$ $\qquad$ $\underbrace{\delta(s,1) = \emptyset}$ ← empty set

interpret as
"always fail" or
"gracefully crash"

try:
  run NFA
catch:
  reject.

type sig of extended transition is $\delta^*: Q \times \Sigma^* \to \mathcal{P}(Q)$

$$\delta^*(s, 01) = \{b\}$$

$\delta^*(s, w)$ might contain states in A
might contain states not in A
might be empty

what does it mean for an NFA to accept a string?

Define an NFA accepting w to mean
at least one state in $\delta^*(s,w)$ is in A
i.e. $\delta^*(s, w) \cap A \neq \emptyset$

Language of NFA N: $L(N) = \{ w \in \Sigma^* \mid \delta^*(s,w) \cap A \neq \emptyset \}$.

Several Interpretations

- Magic fairy that tells you which transition to take at each step to get to an accepting state (if possible)

- Verification: user claiming $w \in L(N)$ has to provide proof in the form of seq of transitions

- Many threads in parallel accept if at least one thread accepts "Many worlds"

Remark:
every DFA can be interpreted as an NFA.

- DFA $M = (Q, \Sigma, \delta, s, A)$
build $N = (Q, \Sigma, \delta', s, A)$ $\quad \delta'(q,a) = \{\delta(q,a)\}$

- purely graphically "see" every DFA is an NFA

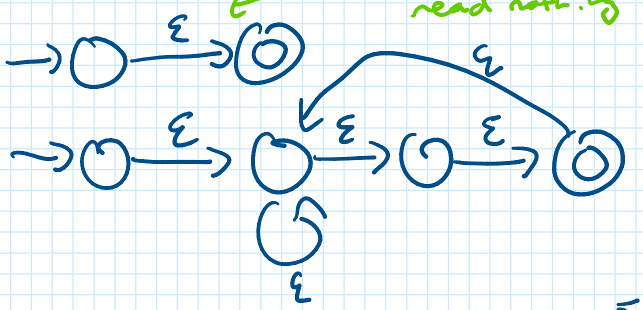
next tuesday: NFA can be converted to a DFA

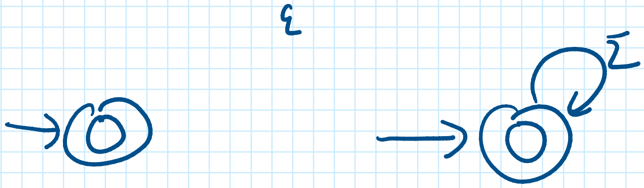
Ex. Given $w = a_1 a_2 a_3 \cdots a_n$ NFA for $\{w\}$

empty string

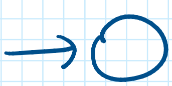
ex. $w = \varepsilon$ NFAs for $\{\varepsilon\}$

"means" read nothing





ex. NFA for ϕ



ε → reject

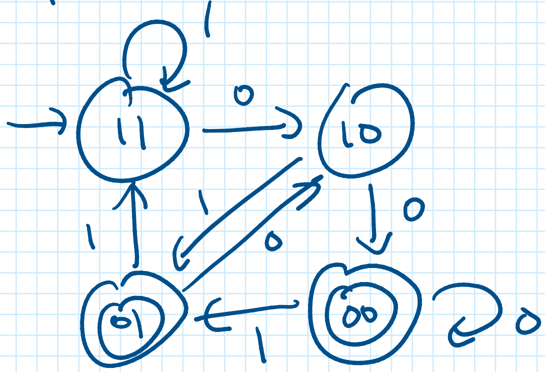
any other string → crash → reject

← DFA needs transition

ex. often (?) NFAs are smaller than DFAs for the same language.

$L = \{ w \mid$ second to last symbol in w is $0 \}$

DFA:



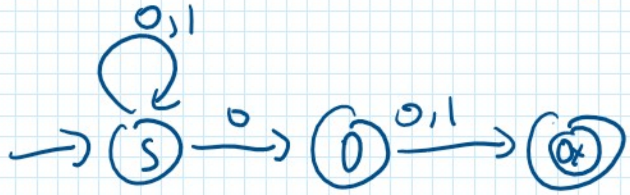
4 states

turns out to be optimal,

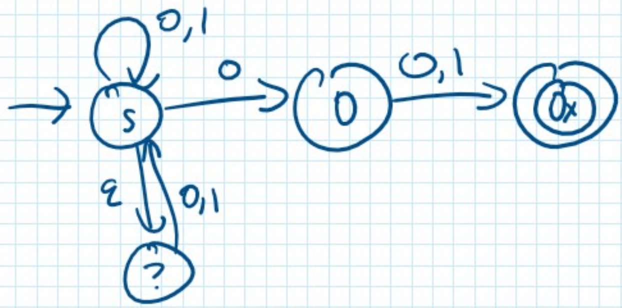
cannot get smaller DFA (why? later)

state label ≡ last two symbols seen (1 ... not 0) state label is last two symbols seen (1 means not 0)

NFA:



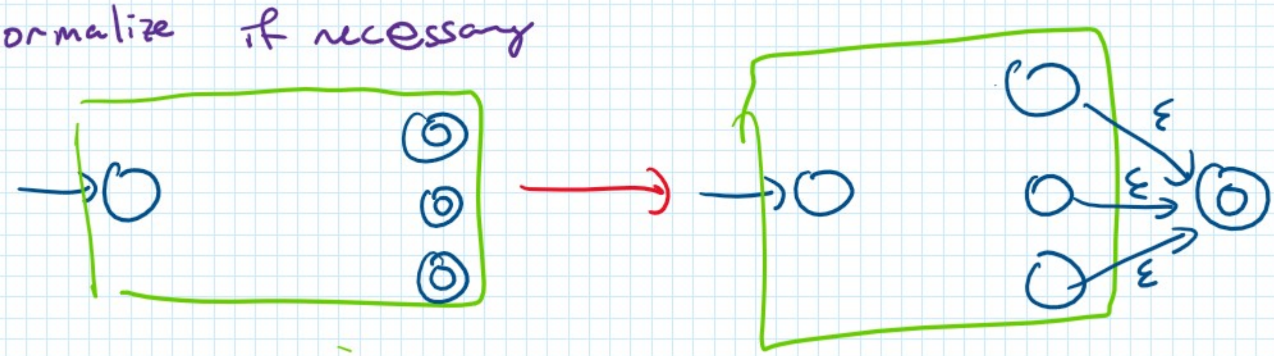
3 states



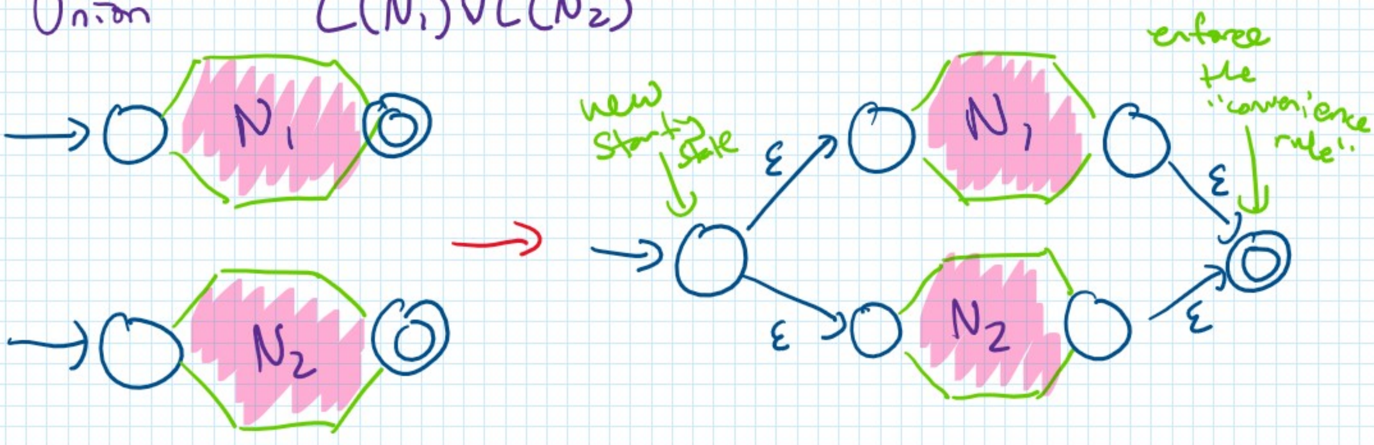
Closure properties of NFAs

For convenience, assume that the NFAs given to us have exactly one accepting state

normalize if necessary

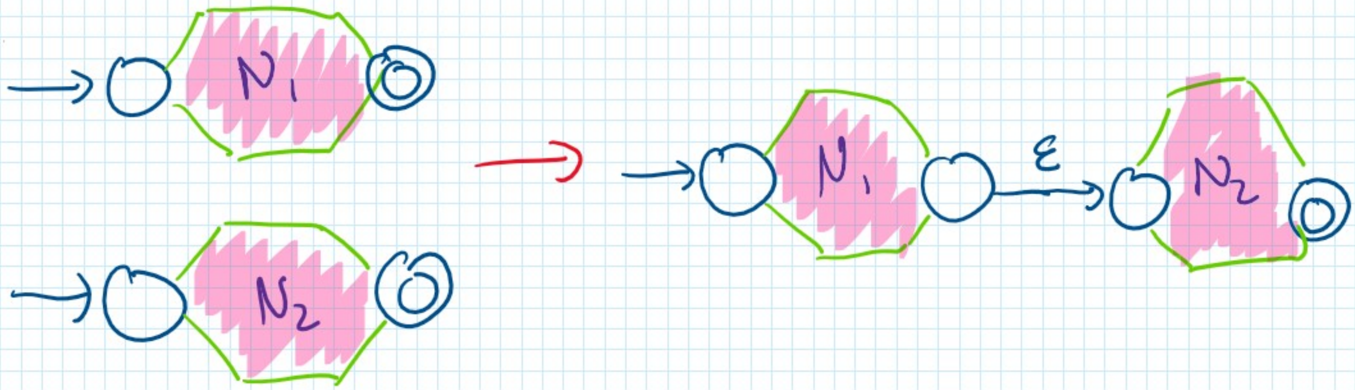


— Union $L(N_1) \cup L(N_2)$

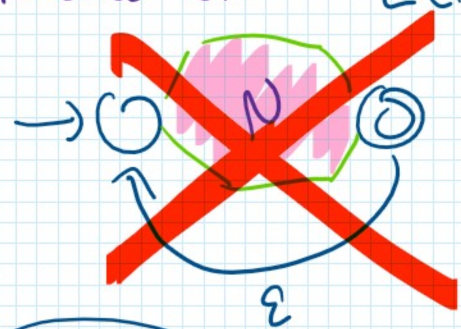


— Concatenation $L(N_1) \cdot L(N_2)$

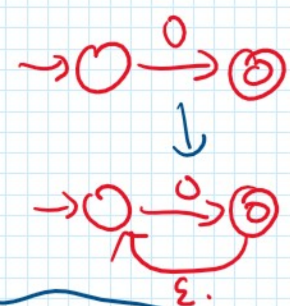
— Concatenation $L(N_1) \cdot L(N_2)$



— Kleene Star $L(N)^*$

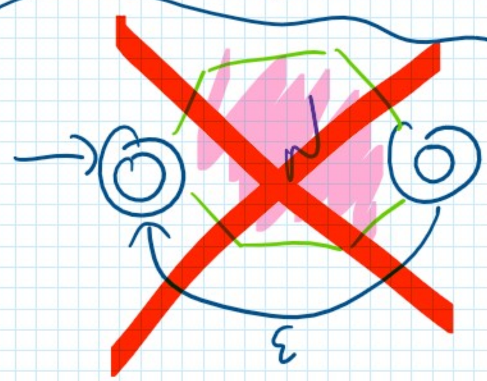


problem: NFA for $\{0\}$

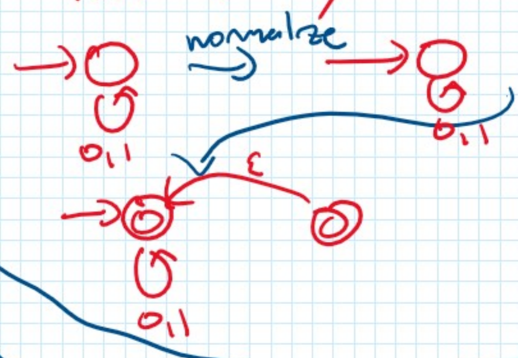


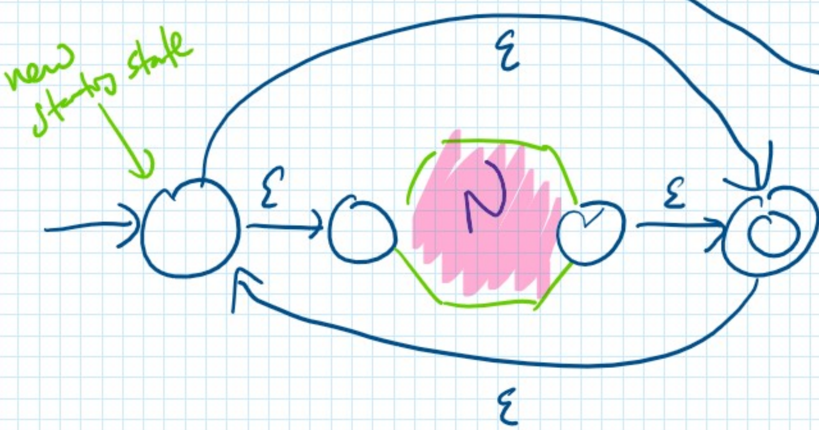
$\varepsilon \in \{0\}^*$

$\varepsilon \notin L(\text{this NFA})$

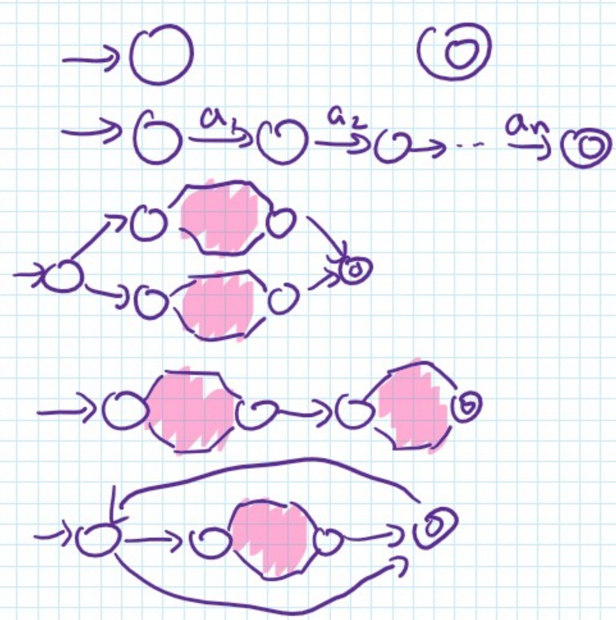


NFA for $\emptyset$





regex	Lang	NFA
ϕ	ϕ	
w	$\{w\}$	
$r_1 + r_2$	$L_1 \cup L_2$	
$r_1 r_2$	$L_1 \cdot L_2$	
r^*	L^*	



Given any regex can construct (recursively) an NFA for the same language

Thompson's algorithm

x