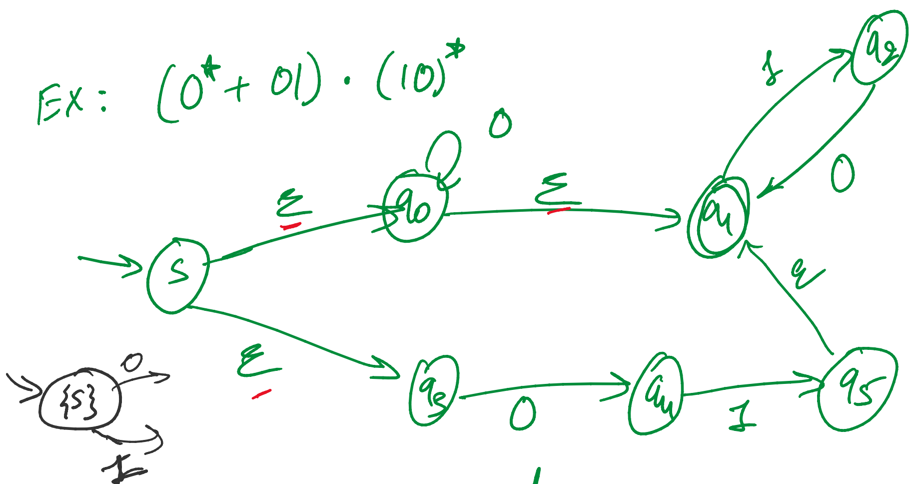
Thursday, February 3, 2022 2:17 PM

Regular Languages $\longleftrightarrow$ DFA Languages

Last lec: $\longrightarrow$ NFA

- allows choices
- " ε-transitions

Ex: $(0^* + 01)\cdot(10)^*$



$T = \{s\}$
$a = 0$

$s \xrightarrow{\varepsilon} q_0 \xrightarrow{0} q_0 \xrightarrow{\varepsilon} q_1$

Thm: If L is regular
Then L is accepted by some NFA M.

Pf: Let r be the reg. exp. for L

$\{+, \cdot, *\}$

Construct M recursively

Base case: $r = \varepsilon$, $r \in \Sigma$

Induction: (See proof from the last lec)

Set of Regular Lang
= " " accepted by NFA
= " " " " DFA

Plan:

Regular $\xrightarrow{\text{Induction/Recursion}}$ NFA $\xrightarrow{1}$ DFA
last lec

★ NFA $\longrightarrow$ DFA.

Thm: If L is accepted by NFA M
Then L " " " some DFA M'

Pf: Constructive : Power-set Construction.

Let NFA $M = (Q, \Sigma, s, \delta, A)$ $\quad \delta : Q \times \{\Sigma \cup \{\varepsilon\}\} \longrightarrow \underline{P(Q)}$

Construct DFA $M' = (Q', \Sigma, s', \delta', A')$

$$Q' = P(Q)$$

$$s' = \varepsilon\text{-reach}(s)$$

$$A' = \{ T \in \underbrace{P(Q)}_{=Q'} \mid T \cap A \neq \emptyset \}$$

$$\delta' : Q' \times \Sigma \longrightarrow Q' = P(Q)$$

$T \in Q'$, $a \in \Sigma$

$$\delta'(T, a) = \bigcup_{q \in T} \delta^*(q, a)$$

Lemma: $\delta'^*(T, x) = \bigcup_{q \in T} \delta^*(q, x)$

Pf: By induction (boring)...

$\delta'^*(s', x) \in A'$

Pf. By Induction (...)

Then, $x \in L(M') \iff \delta'^*(s', x) \in A'$

$\iff \delta'^*(\varepsilon\text{-reach}\{s\}, x) \in A'$

(By Lemma) $\iff \bigcup_{q \in \varepsilon\text{-reach}\{s\}} \delta^*(q, x) \in A'$

(By def δ^* for an NFA) $\iff \delta^*(s, x) \in A'$

(By def A') $\iff \delta^*(s, x) \cap A \neq \emptyset$

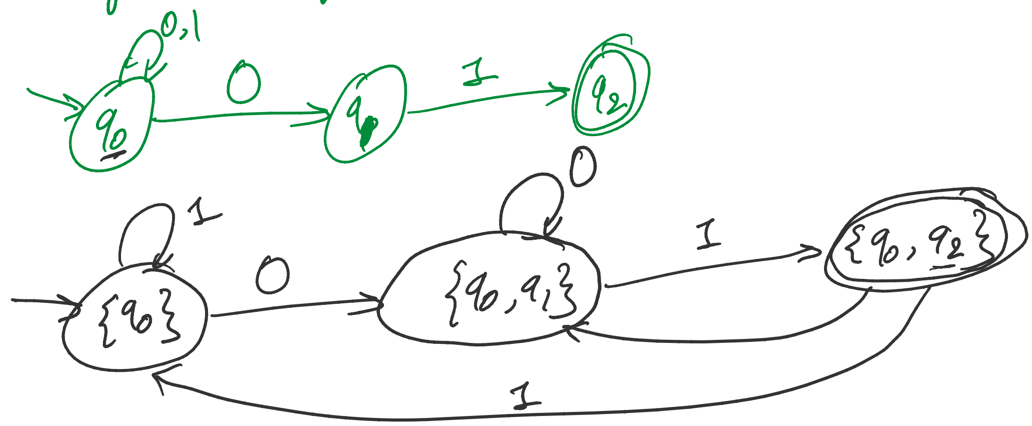
$\underset{\text{accept in NFA.}}{\iff} x \in L(M)$ ∎

Ex 1: Strings ending w/ 01.

NFA

↓

DFA



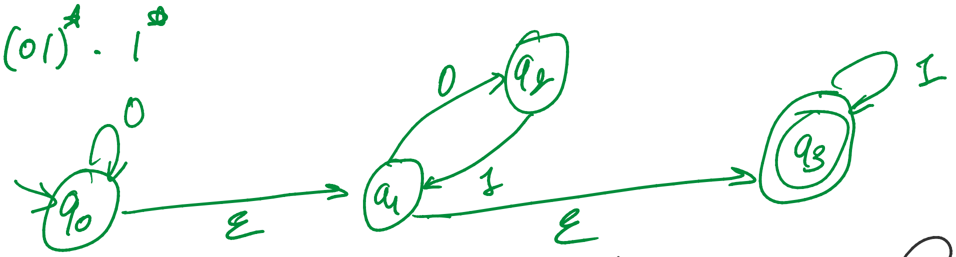
$\delta'(\{q_0\}, 0) = \delta^*(q_0, 0) = \delta(q_0, 0)$
$= \{q_0, q_1\}$

$\delta'(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$
$= \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

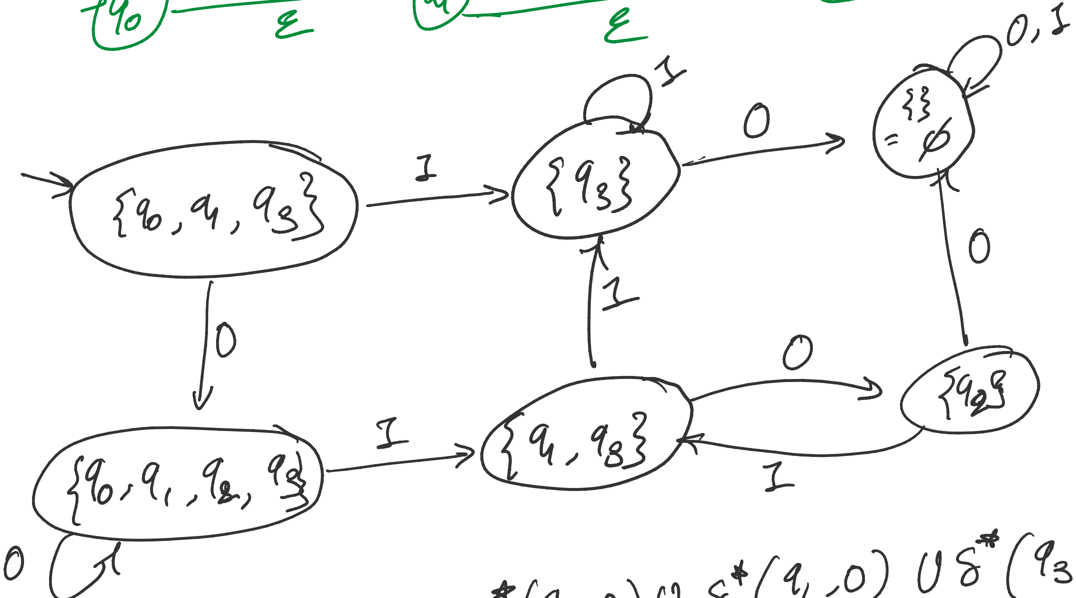
$\delta'(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$
$= \{q_0, q_1\} \cup \{\}$
$= \{q_0, q_1\}$

Ex 2: $0^* \cdot (01)^* \cdot 1^*$

Ex 2 : $0^* \cdot (01)^* \cdot 1^*$



NFA
↓
DFA



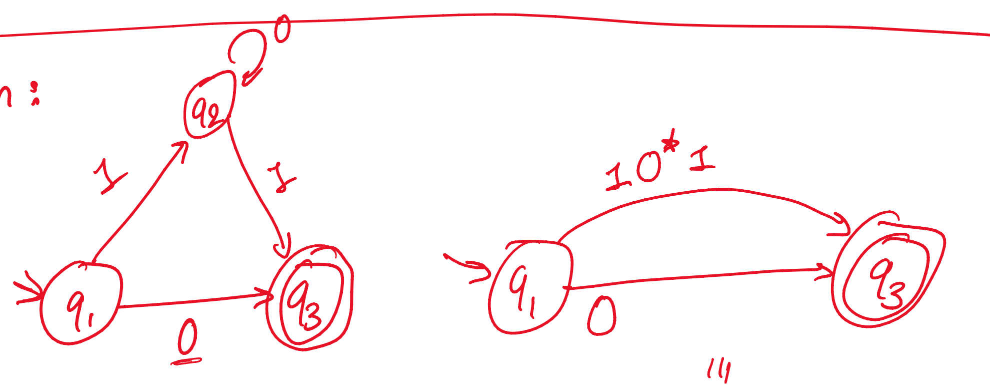
$$\delta'(\{q_0, q_1, q_3\}, 0) = \delta^*(q_0, 0) \cup \delta^*(q_1, 0) \cup \delta^*(q_3, 0)$$
$$= \{q_0, q_1, q_2, q_3\} \cup \{q_2\} \cup \{\}$$
$$= \{q_0, q_1, q_2, q_3\}$$

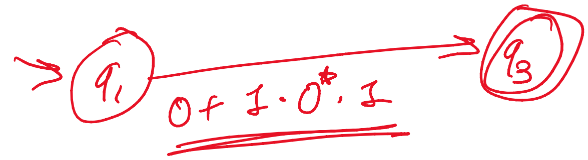
★ DFA ⟶ Regular lang.

Thm: If L is accepted by DFA ($\to$ NFA) M
Then L is regular.

Pf: Constructive. "State Elimination Method"

Intuition:



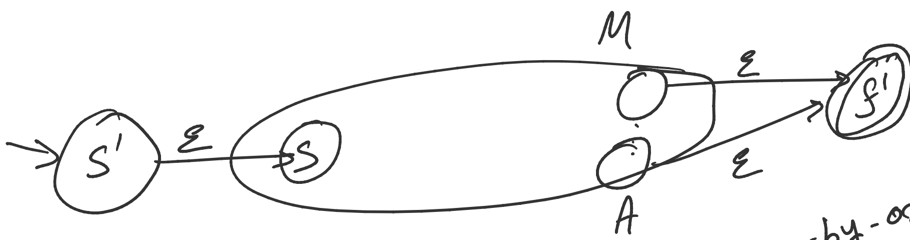


Pf Sketch Let $\overset{DFA}{M} = (Q, \Sigma, \delta, s, A)$

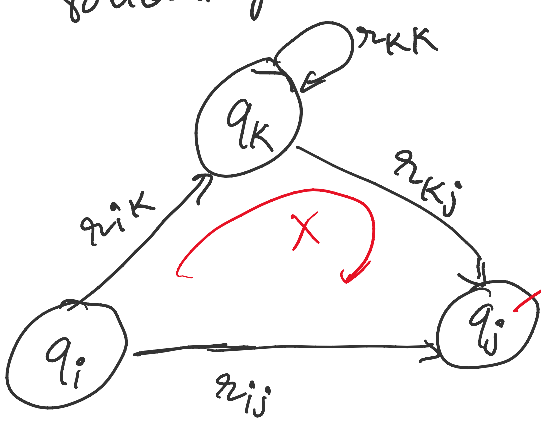
Construct reg. exp. for language $L = L(M)$

By the following method.

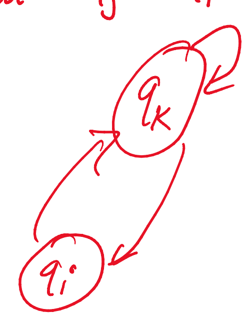
1. Introduce s' & f' states
   Add edge from s' to s on ε
   " " " every state in A to f' on ε.



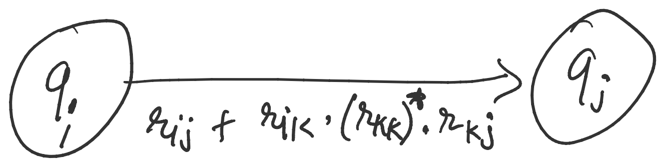
2. Eliminate states of Q one-by-one through the following rule



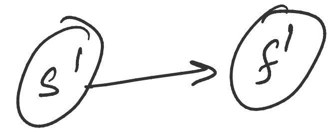
It may be that $q_j = q_i$



$\downarrow$

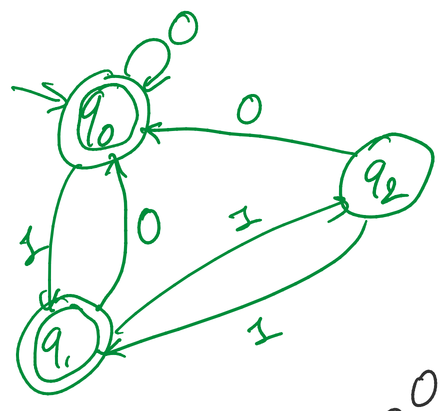


3. Return label on edge

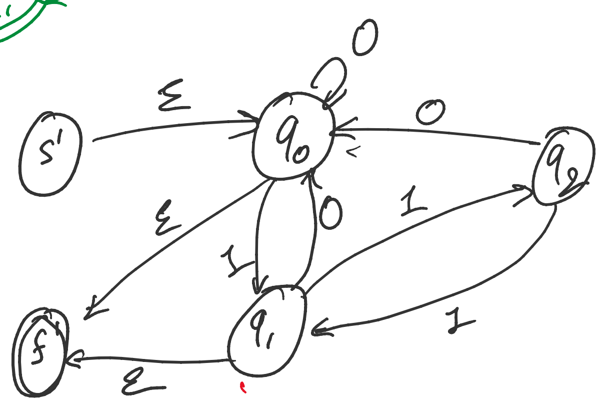


Ex:

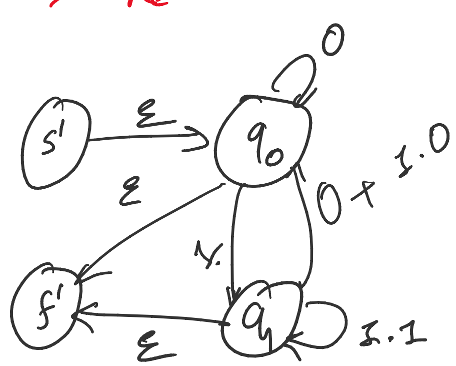
DFA → Reg. Exp.



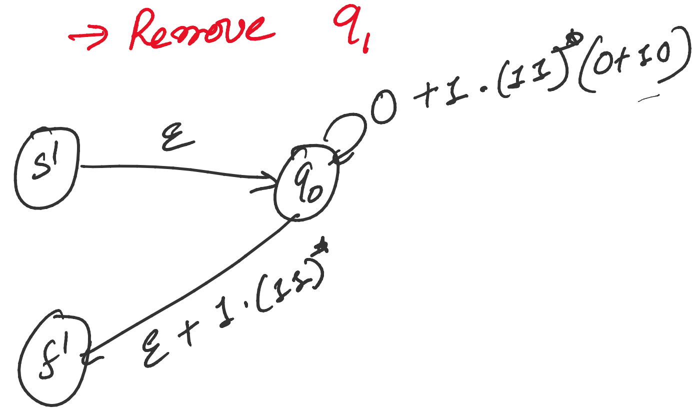
1.



2. → Remove q_2



→ Remove q_1



→ Remove q_0

$$S' \xrightarrow{(0+1\cdot(11)^*(0+10))^* \cdot (\varepsilon+1\cdot(11)^*)} S'$$

Kleene's Thm (1956):

L is regular iff L is accepted by some DFA M.

Pf: Constructive / Algorithm

$$\begin{tikzcd}
\text{Regular} \arrow[r, "\text{Inductive/}", "\text{Recursive}"'] & \text{NFA} \arrow[r, "\text{Power-set}", "\text{Construction}"'] & \text{DFA} \arrow[ll, bend left, "\text{State Elimination Method}"]
\end{tikzcd}$$

Cor: If L is regular then $\bar{L}$ is regular

If L_1, L_2 are " " $L_1 \cap L_2$ "

$L_1 \oplus L_2$ "

⋮

Remark: Closed under other ops, like

Reverse, Homomorphism

Prefix, Suffix, Subseq, Supseq ...

All Regular Languages ?

$\mathcal{L} = \{$ set of all Regular Languages $\}$

$\mathcal{L}$ is "closed under op" f if

$\forall L \in \mathcal{L}$, $f(L) \in \mathcal{L}$