CPSC 421/501 Oct 7, 2024

- Motivation: $\{a^3, a^5\}^*$, L^*

- DFA: $\delta: Q \times \Sigma \to Q$

  NFA: $\delta: Q \times \Sigma_\varepsilon \to \text{Power}(Q)$

- NFA's $\longrightarrow$ DFA's

- Thm: If L_1, L_2 ~~are~~ are regular,

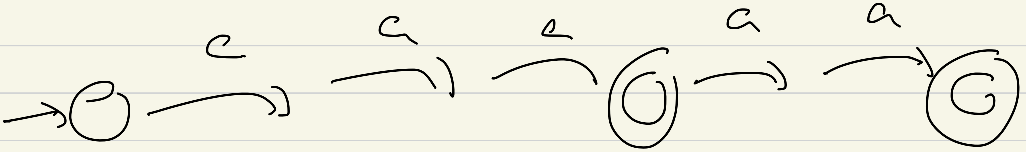
then so are $L_1 \cup L_2$, $L_1 \circ L_2$, L_1^*.

- Regular Expressions:

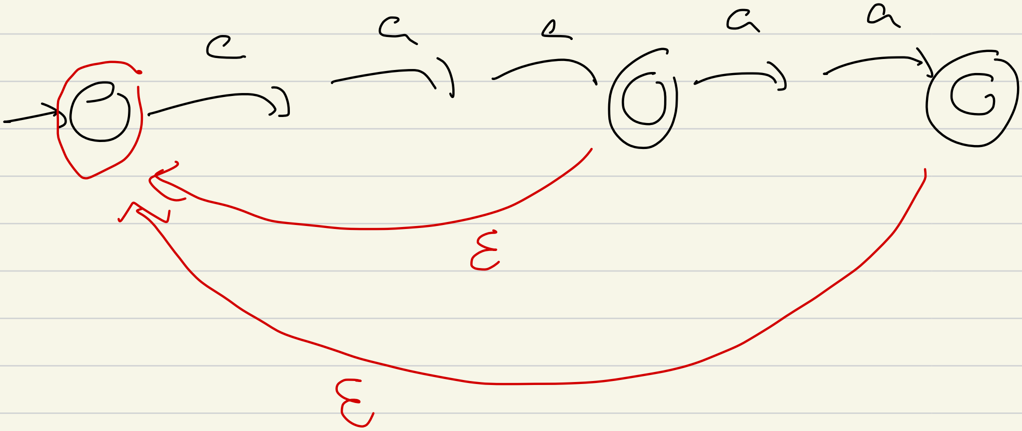
  $\emptyset$, subset of Σ_ε, $\cup$, $\circ$, $*$, $($ $)$

- DIV-BY-3 (warning)

$\{a^3, a^5\}$



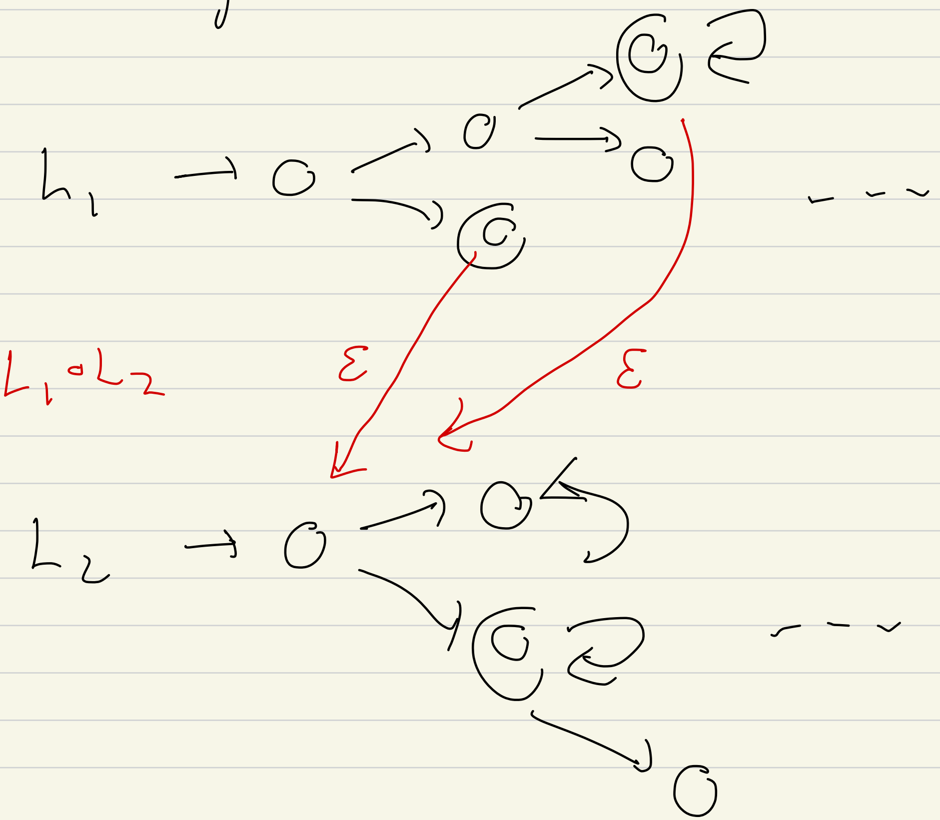
$\{a^3, a^5\}^*$



Say L_1, L_2 are regular,

$$L_1 \circ L_2 = \{ s_1 \circ s_2 \mid s_1 \in L_1, s_2 \in L_2 \}$$

is regular



The key $L \to L^*$

or $L_1, L_2 \to L_1 \circ L_2$

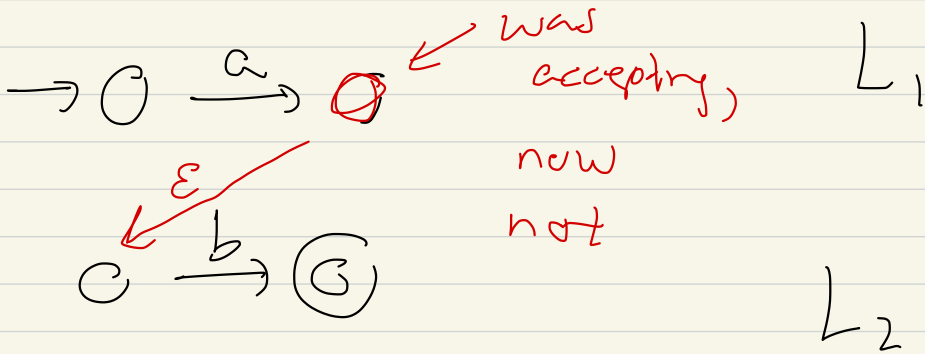
is in a non-deterministic algorithm (finite automata, Python, T.M. Stack)

we accept a string iff there is at least one computation path that leads to an accepting state.

Rem:

$L_1 = \{a\}, \quad L_2 = \{b\}$

$L_1 \circ L_2 = \{ab\}$



Recipe: We leave L_1, L_2 NFA's alone, except have an ε (jump) from each final state of L_1 to initial state of L_2

Plus:

- We make every accepting state of L_1 non-accepting

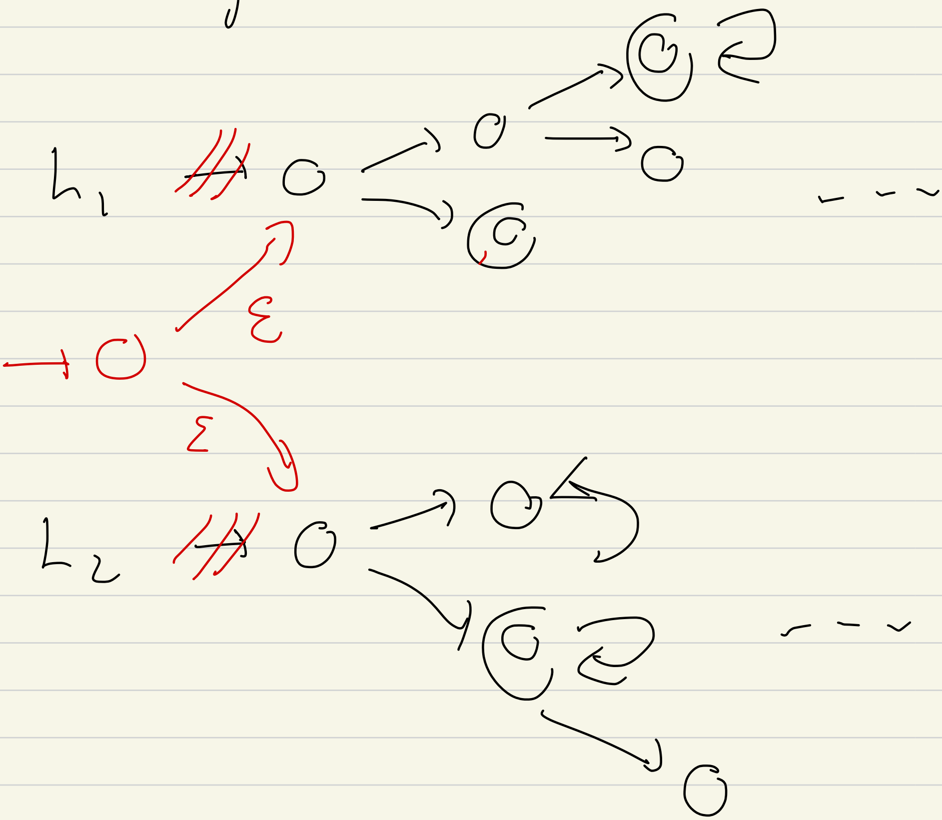
- We make the initial state of L_2 no longer initial

[Thanks to Kevin Liu for this page of modifications]

C.H.

How about $L_1 \cup L_2$:

is regular

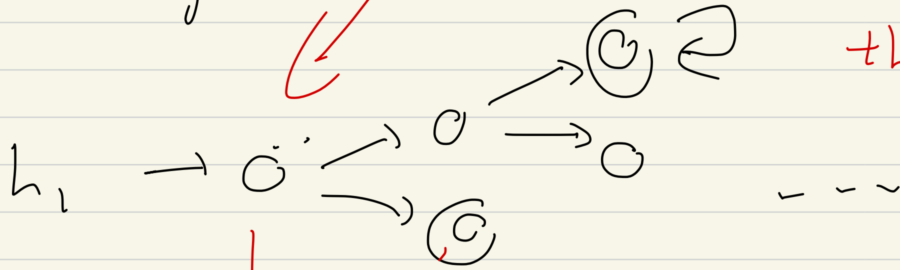


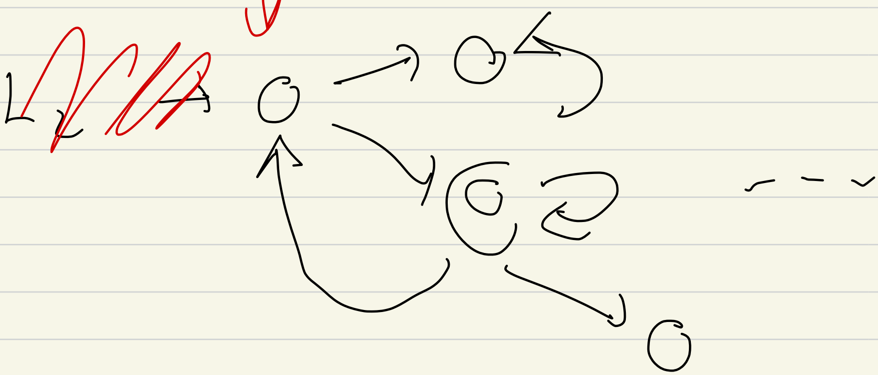
New initial state, ε jumps

from " " " to initial states L_1, L_2

Rem: If no way to return to initial state of L_1, then

is regular

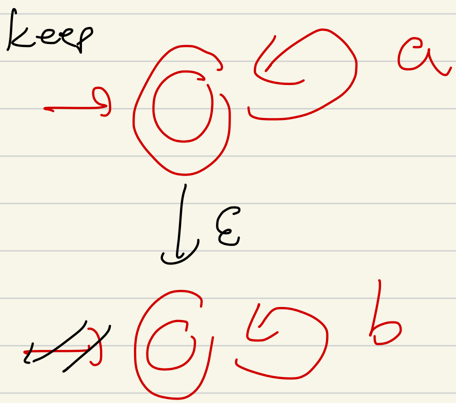




e.g., $L_1 = \{a\}^*$ $L_2 = \{b\}^*$

$L_1 \cup L_2 = \{a\}^* \cup \{b\}^*$

so $ab \notin L_1 \cup L_2$



L_1

L_2

NFA accepts $a^* b^* = L_1 \circ L_2$

not $L_1 \cup L_2 = (a^*) \cup (b^*)$

Regular Expressions §1.3 [S16p]

A regular expression over Σ is:

- $\emptyset$
- ε
- a symbol in Σ
- anything you can get from and taking

$\cup$, $\circ$, $*$

Examples: $a^* = (a)^*$

$a^* b^*$, $(a^*) \cup (b^*)$,

$\emptyset$, ε, $(ab)^* \cup (a^2 b)^*$

$\uparrow$

$a^2 = aa$

We say, a regular expression <u>describes</u> a language:

$\emptyset \longrightarrow \emptyset$

$\varepsilon \longrightarrow \{\varepsilon\}$

$a \rightarrow \{a\}$

$R_1 \cup R_2 \longrightarrow$ (language described by R_1) $\cup$ (lang. desc. by R_2)

Similarly

$R_1 \circ R_2$ - - - - ( )

$\circ$ ( )

$R_1^* = (\quad)^*$

We could add $\neg$ (negation), $\cap$, $+$

$R^+ = (\text{lang descr by } R)^+$

$L^+ = L \circ L^*$

$\quad = L^1 \cup L^2 \cup L^3 \cup \ldots$

Thm: Any regular language
$\Rightarrow$
is described by a regular expression, and any language described by a regular expression
$\Rightarrow$
is regular.

So we should:

finish NFA $\xrightarrow{\text{equivalent}}$ DFA

We might prove the theorem,

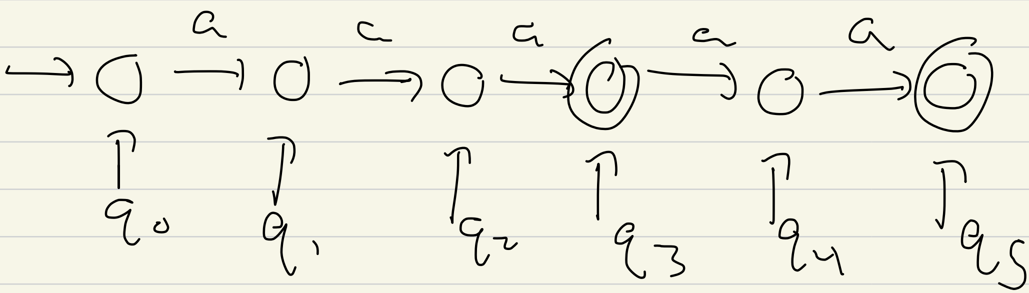
but we won't ... We'll

prove 1/2 the theorem ...

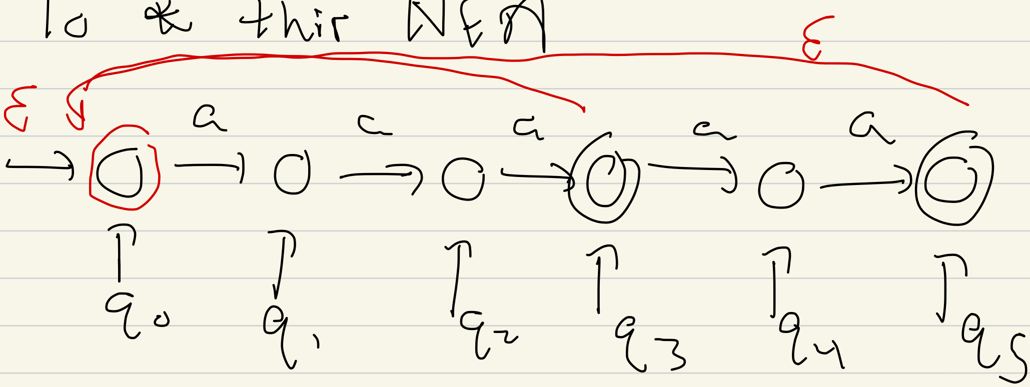
NFA → DFA

$\{a^3, a^5\}^*$

NFA



To * this NFA



$$\text{Power}(Q) = \text{Power}(\{q_0, q_1, \dots, q_5\})$$

$\emptyset$

$\{q_0\}$ $\{q_1\}$

$$\{q_2, q_3, q_5\} \xrightarrow{a} \{q_3, q_0, q_4, q_1\}$$

all inputs that can arrive in q_2, q_3, q_5 but only these

Given NFA: $(Q, \Sigma, \delta, q_0, F)$

$$\delta : Q \times \Sigma_\varepsilon \longrightarrow \mathrm{Power}(Q)$$

Let DFA have state set

$$Q' = \mathrm{Power}(Q)$$

we form $\delta : \mathrm{Power}(Q) \times \Sigma \longrightarrow \mathrm{Power}(Q)$

$q_0' = \{ q_0,$ anything else we can reach from q_0 in NFA with ε jumps $\}$