CPSC 421/501, Oct 1, 2020

Section 1.2 [Sip] NFA's:

- NFA example (with $\delta(q,a) = \emptyset$)

- Thm: L recognized by an NFA $\Longrightarrow$
  " " " some DFA.

- Corollary: Regular languages closed under $\cap$, $\cup$, complement, $\circ$, $*$

Section 1.3 [Sip] Regular Expressions:

- Def of Regular Expression: any $\cup$, $\circ$, $*$ of {single letter}, $\{\varepsilon\}$, $\emptyset$

- Thm: L described by a regular expression $\Longleftrightarrow$ L is regular.

[Some searching for reg. exp. in strings is to actually build NFA/DFA.]

Breakout Room Questions:

(1) How many states needed to recognize $\{a^5, a^7\}^*$ by a DFA

seen before

(2) How many states needed to recognize $\{a^5, a^7\}^*$ by an NFA

seen before

(3) If an NFA has 1000 states, its corresponding DFA may have roughly 2^{1000} states. Is there a relatively quick way to see if the NFA accepts a given string?

(4) Give an NFA that recognizes

$$L = \{ w \in \{0,1\}^* \mid \text{the } 3^{rd} \text{ to last symbol of } w \text{ is } 1 \}$$

$$= \{0,1\}^* \circ \{1\} \circ \{0,1\}^2$$

fundamental

$$= \{ \sigma_1 \ldots \sigma_k \mid k \geq 3, \ \sigma_1, \ldots, \sigma_k \in \{0,1\} \text{ with } \sigma_{k-2} = 1 \}$$

(5) Give a DFA that recognizes L in question (4)

Remark: Same bonus question for HW #2 can be solved and submitted for HW #3, but only counted once.

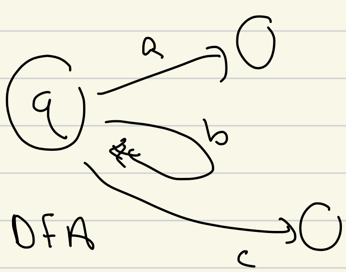
Bonus question: Last years I gave bonus questions for zero credit. These are beyond material.

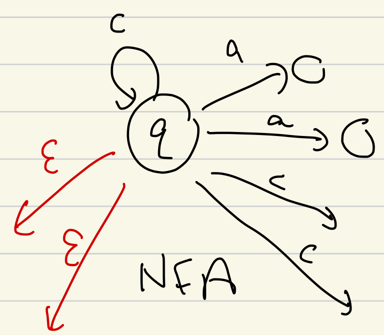
And # submissions was 0.

This year, bonus questions addition 10%,

so this can add between 0 and 1% to your total grade.

NFA is like a DFA but (say $\Sigma = \{a,b,c\}$)





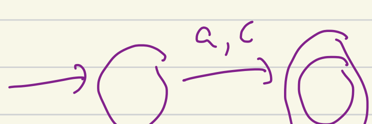
could have no b arrows

"jump" "reading the empty string, ε"

NFA, N = non-deterministic: A string is accepted by an NFA if there is at least one way to go thru NFA and get to an accepting state.

---

e.g. $\Sigma = \{a,b,c\}$

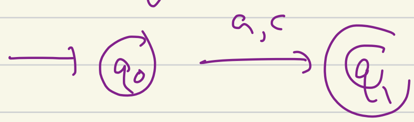


recognizes

$L = \{a, c\}$

e.g. $w = abb$ as input

read ~~aa~~ a

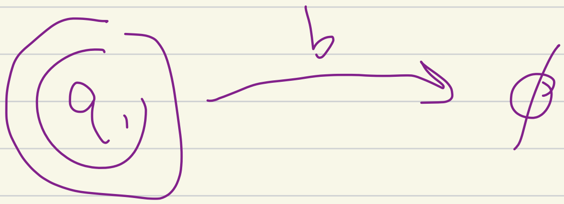


start

read the first a

then you read b but there are no b arrow

each possible path, you can have



computation stops and fails

Why NFA's?

$L_1 = \{a^3\}^* = \{aaa\}^*$ ← star operation over a language

$\left(\text{rem: } \Sigma^* = \text{words over } * \right.$

$\left. \quad = \Sigma \text{ with star operation} \right)$

$= \{a^k \mid 3 \text{ divides } k\}$

$L_2 = \{a^5\}^* = \{a^l \mid 5 \text{ divides } l\}$

$L_1 \circ L_2 = \left\{ a^{k+l} \;\middle|\; \begin{array}{l} 3 \text{ divides } k \\ 5 \text{ divides } l \end{array} \right\}, \quad \Sigma = \{a\}$

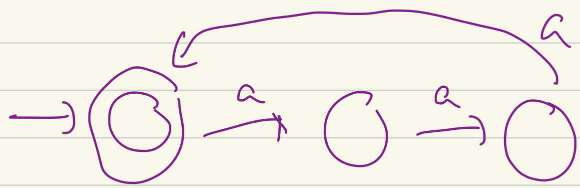
$aaaa\ldots a = a^{13} = \{a^3\}^* \circ \{a^5\}^*$

start to read somes a's

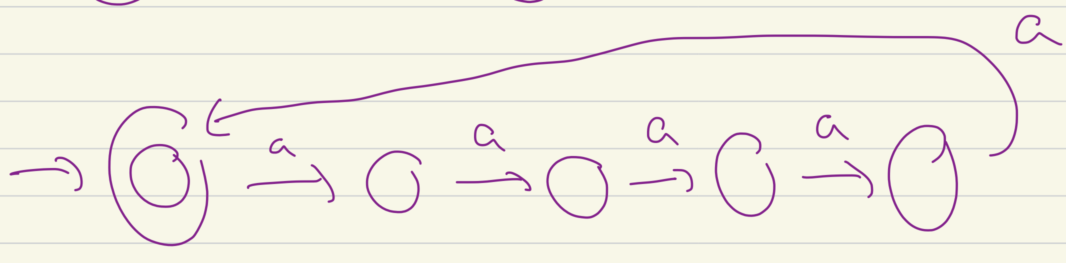
when do you go from L_1 to L_2

NFA

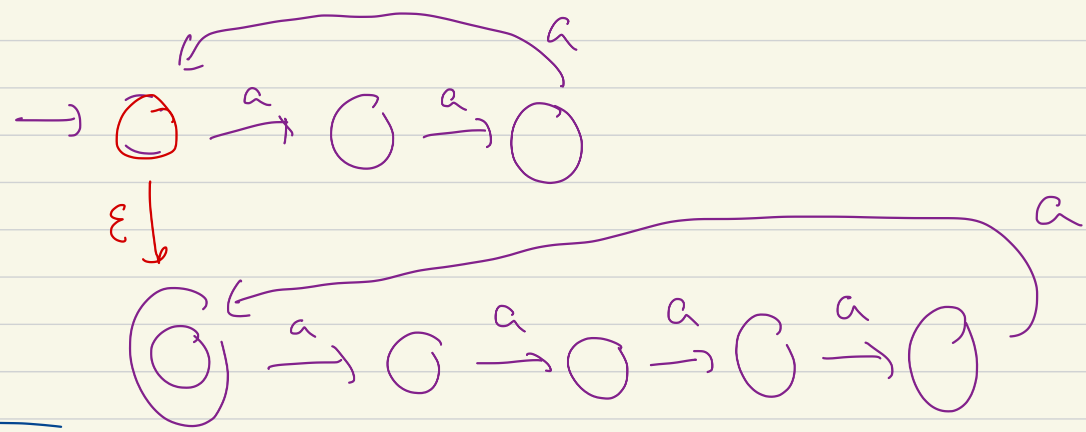
DFA for L_1



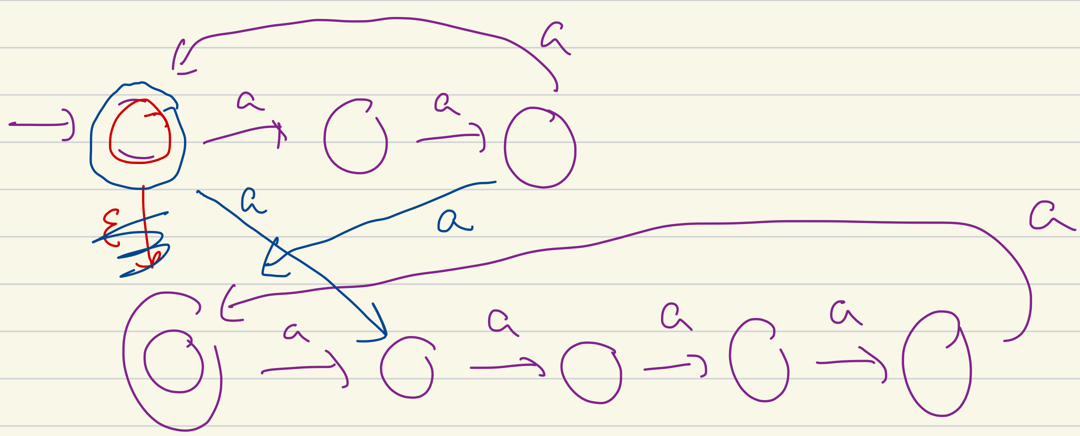
DFA for L_2



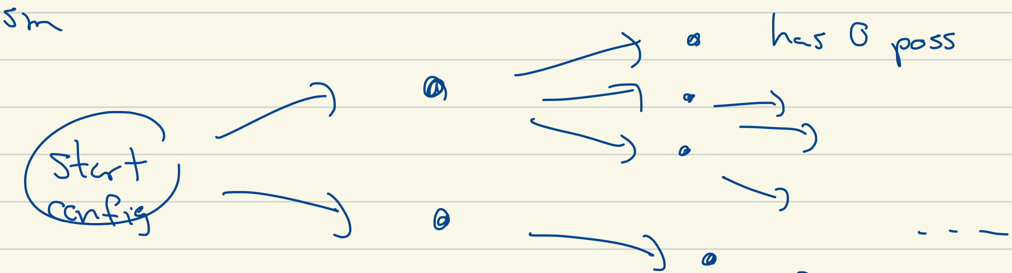
NFA for $L_1 \circ L_2$



Without ε jump



non-determinism



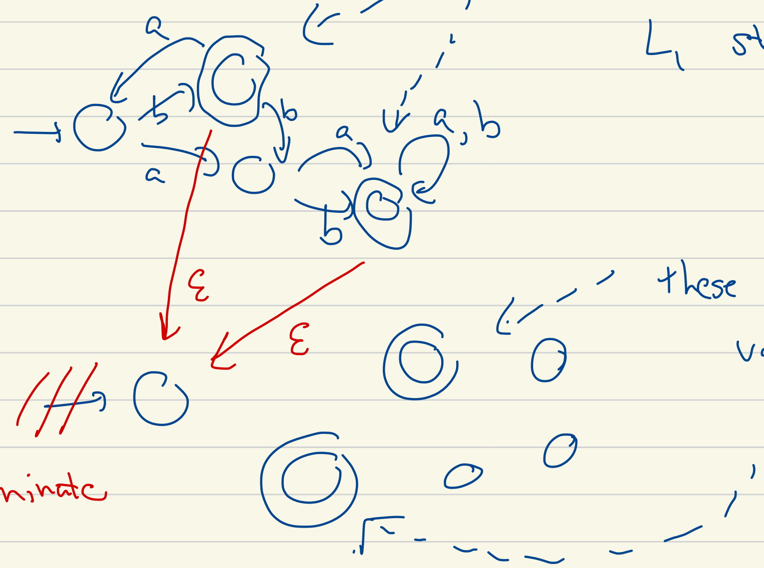
2 possible next steps

Thm: If L_1, L_2 are regular, i.e. there are DFA's recognizing L_1, L_2, then $L_1 \circ L_2$ is regular.

Procedure

DFA for L_1

DFA for L_2



these are valid L_1 strings

these are valid L_2 strings

eliminate

NFA for $L_1 \circ L_2$

Procedure : (1) jump from ◎ in first machine to → ○ in second

(2) eliminate → ○ in second

Now DFA for L_1, DFA for L_2 $\rightsquigarrow$ NFA for $L_1 \circ L_2$.

Observe! A DFA is an NFA.

And for any NFA, there is an equivalent DFA, i.e. if an NFA recognizes a language, L, then there is a DFA recognizing L.

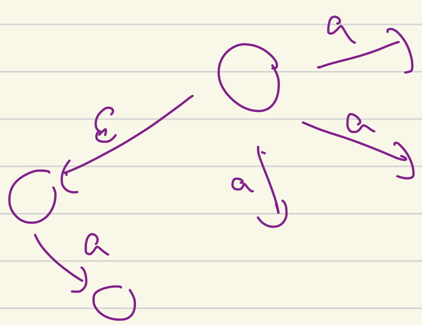
Rem: P : polytime algorithms,

NP = non-deterministic polytime algorithms

$P \overset{?}{=} NP$

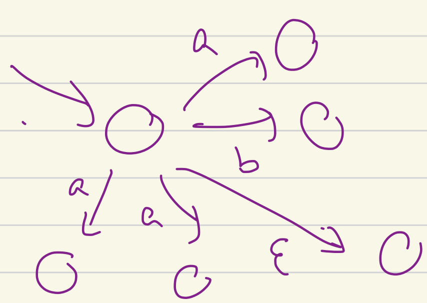
\$ 10^6

NFA allows a "look ahead" idea



How to convert an NFA to an equivalent DFA:

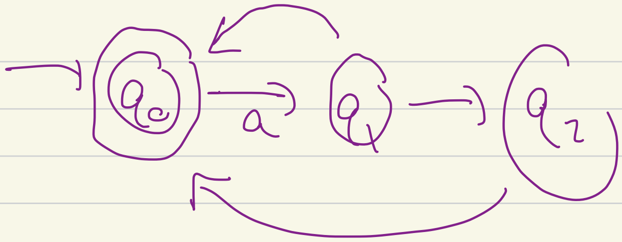
NFA :



$Q = \{q_0, q_1, \dots, q_5\}$

Idea: State set of DFA = Power(Q)

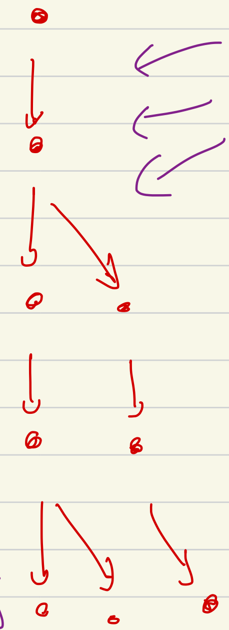
q_0, q_1, q_5 $\xleftrightarrow{\text{meaning}}$ When considering the NFA, $\{q_0, q_1, q_5\}$ are all possible states e.g.



NFA for

$\{a^2, a^3\}^*$

non-deterministic comp tree



Idea: Initially $\{q_0\}$ ①

Reading a: $\{q_1\}$ ②

Reading next a: $\{q_0, q_2\}$ ③

" " a: $\{q_1, q_0\}$ ④

" " " : $\{q_0, q_2, q_1\}$ ⑤

← list all possible states

Problem: If NFA has m states $= |Q|$ Q for NFA

DFA has 2^m states $= |\text{Power}(Q)|$

Remark: States of DFA

once we're at $\{q_0, q_2, q_1\} = Q$

each "a" takes $Q \to Q$

Really have only 5 reachable states

Breakout rooms: I suggest

3, then 1 or 4
2 or 5

and make sure that you're comfortable with NFA's, DFA's and taking NFA → DFA

[10:27 → 10:37]

Question 3 (Is this in [Sip] textbook?)

NFA has states $q_0, q_1, \ldots, q_{999}$;

Can we "implement" the corresponding DFA with a practical algorithm?

DFA states are subsets of Q:

can be there, or can't

$\_\_\ \_\_\ \ldots\ \_\_$

$q_0\ q_1\ \ldots\ q_{999}$

example: can, can't, can, can, ..., can't ← {can, can't}

1 0 1 1 ... 0 ← 0/1

read σ

1 1 1 1

state (1000 bits)

go thru the NFA → next symbol read

(1000 bits)

Solution: If NFA has m states

DFA state $\longleftrightarrow$ m-bits $\begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \leftrightarrow \begin{Bmatrix} \text{can't} \\ \text{can} \end{Bmatrix}$

There is an algorithm for each symbol of input take polytime in # NFA states, m.

(Don't generate a table of 2^m states + transitions)

Breakout room ① & ② $\{a^5, a^7\}^*$

$= \{ \cdots \}$? not easy...

Breakout room ④ & ⑤ we will discuss these when we cover non-regular languages & min # states of DFA for a given language.

NFA's + DFA's !

If L_1, L_2 regular $\Rightarrow$

$L_1 \circ L_2$, L_1^* , $L_1 \cup L_2$ is regular

$\Rightarrow$ any "regular expression" $\leftarrow$ $\{1,3\}$

e.g. $\{a,c\}^* bb \{ac, bc\}$ $[S_{1p}]$

represents a language recognizable

by an NFA