Section 1.2 [Sip] NFA’s:

- NFA example (with δ(q,a) = ∅)
- Thm: L recognized by an NFA ⇒ Some DFA

- Corollary: Regular languages closed under ∅, U, complement, ∅, *

Section 1.3 [Sip] Regular Expressions:

- Def of Regular Expression: any U, ∅, *
  of \{ single letter \}, \{ ε \}, ∅

- Thm: L described by a regular expression
  \iff 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?

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

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

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

\[ = \{0,1\}^* \cup \{1\} \cup \{0,1\}^2 \]

Fundamental = \{ \sigma_1 \ldots \sigma_k \mid k \geq 3, \sigma_1, \ldots, \sigma_k \in \{0,1\} \}

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 year 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 $\Sigma = \{a, b, c\}$)

\[
\begin{array}{c}
\text{DFA} \\
\begin{array}{c}
\circ \\
\rightarrow b \\
\downarrow c \\
\circ \\
\end{array}
\end{array}
\quad \begin{array}{c}
\text{NFA} \\
\begin{array}{c}
\circ \\
\rightarrow a, b, c \\
\quad e \\
\end{array}
\end{array}
\]

could have no b arrows

"jump" "reading the empty string, $e$"

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

\[
\begin{array}{c}
\circ \\
\rightarrow a, c \\
\downarrow \\
\circ \\
\end{array}
\]

recognizes $L = \{a, c\}$
e.g. \( w = a b b \) as input

\[
\begin{aligned}
\rightarrow q_0 & \xrightarrow{a,c} q_1 \\
& \text{then you read } b \\
& \text{start read the first a but there are no b arrow}
\end{aligned}
\]

each possible path, you can have

\[
\begin{aligned}
\circ q_1 & \xrightarrow{b} \emptyset \\
& \text{Computation stops and fails}
\end{aligned}
\]

Why NFA's?

\[
\begin{aligned}
L_1 &= \{a^3\}^* = \{a a a\}^* \\
& \text{star operation over a language} \\
\text{(rem: } \Sigma^* = \text{words are +}) &= \{a^k \mid 3 \text{ divides } k\} \\
\text{=} & \Sigma \text{ with star operation}
\end{aligned}
\]

\[
L_2 &= \{a^5\}^* = \{a^l \mid 5 \text{ divides } l\}
\]

\[
L_1 \circ L_2 = \{a^{k+l} \mid 3 \text{ divides } k, 5 \text{ divides } l\}, \Sigma = \{a\}
\]

\[
\text{aaaaa}_a = a^{13} = \{a^3\}^* \cup \{a^5\}^*
\]

// start to read some 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 $\varepsilon$ jump

Non-determinism

Start config

2 possible next steps
Then! If $L_1, L_2$ are regular, i.e. there are DFA's recognizing $L_1, L_2$, then $L_1 \cdot L_2$ is regular.

Procedure:

1. DFA for $L_1$ 
2. DFA for $L_2$ 
3. NFA for $L_1 \cdot L_2$ 
4. Procedure: (1) jump from $\epsilon$ in first machine to $\epsilon$ in second 
5. (2) eliminate $\epsilon$ in second

Now DFA for $L_1$, DFA for $L_2 \Rightarrow$ NFA for $L_1 \cdot L_2$.

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, $NP = \text{non-deterministic polytime algorithms}$

NFA allows a "look ahead" idea

How to convert an NFA to an equivalent DFA:

\[
\begin{align*}
NFA: &\quad Q = \{ q_0, q_1, \ldots, q_5 \} \\
DFA: &\quad \text{State set of DFA} = \text{Power}(Q) \\
\end{align*}
\]

Idea: When considering the NFA, $\{ q_0, q_1, q_5 \}$ are 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" leads $Q \mapsto Q$
Really have only 5 reachable states
Breakout rooms? I suggest

③, then ② or ④

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 \( q_0, q_1, \ldots, q_{aaa} \).

Can we “implement” the corresponding DFA with a practical algorithm?

DFA states are subsets of \( Q \):

- Can be there, or can’t
- \( q_0, q_1, q_{aaa} \)

Example: can’t can’t can can’t → can’t \( \subseteq \{ \) can, can’t \( \} \)

\( \begin{array}{cccc}
\text{read } a & \leftarrow & \text{move } & \text{symbol} \\
\text{over } & & \text{the } & \text{NFA} \\
\text{go } & \downarrow & \text{to } & \text{next symbol} \\
\text{read } & \downarrow & 0, 1 & \text{(1000 bits)} \\
\text{state } & \downarrow & \text{0} & \text{(1000 bits)} \\
\end{array} \)
Solution: If NFA has m states

DFA state $\mapsto m$-bits $\{c\} \mapsto \{can\+\}

There is an algorithm for each symbol of input that runs in polytime in $\#$NFA states, m.

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

Breakout room $\circled{1} \& \circled{2}$ $\{a^5, a^7\}^*$

$= \{\ldots\}$ not easy...

Breakout room $\circled{4} \& \circled{5}$ we will discuss these when we cover non-regular languages & min #states of DFA for a given language.
NEA's + DFA's!

If $L_1, L_2$ regular $\implies$

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

$\implies$ any "regular expression" $\iff \S 1.3$

e.g. $\{a, c\}^* b b \{cc, bc\}$

represents a language recognizable by an NFA