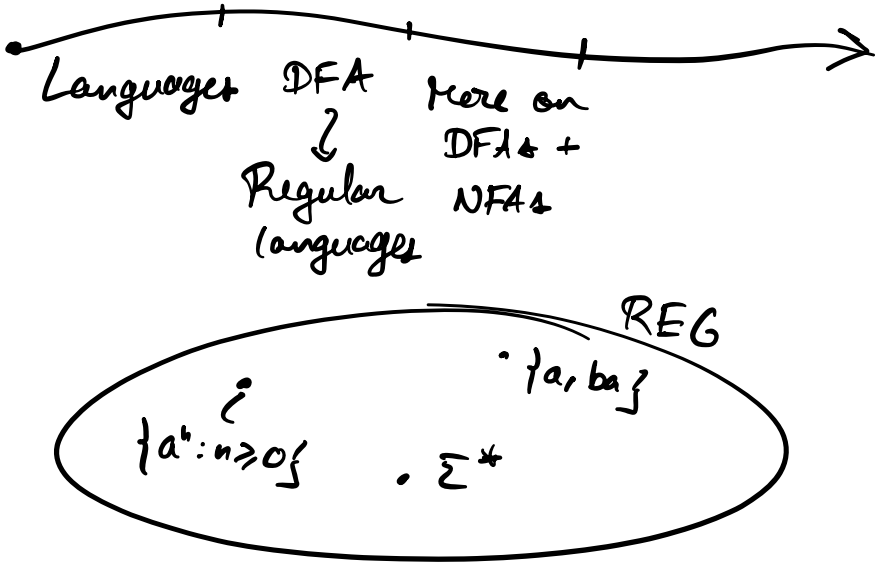
Lecture 4 - Comp 330 - September 12th 2023

Admin

→ Induction

→ Less stressed.



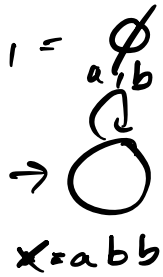
Exercise (Warm-up) $\Sigma = \{a, b\}$. Which of the following languages are regular?

$L_1 = \emptyset$

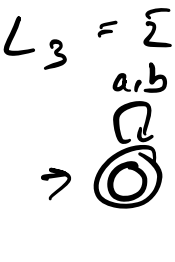
$L_2 = \{\varepsilon\}$

$L_3 = \Sigma^*$

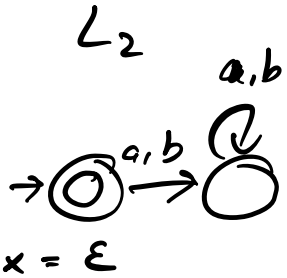
$L_1 = \emptyset$



$L_3 = \Sigma^*$



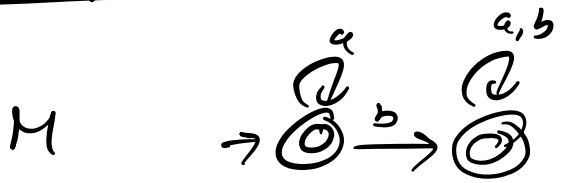
L_2



Recall When designing DFAs, states should encode information

↳ If we don't carefully design DFA, may produce redundant states

For sample $\Sigma = \{a, b\}$

M 

$L(M) = \Sigma^*$

M has redundancy $\Rightarrow$ M is not **minimal**

Def (Minimal DFA) Given a DFA M, we say that M is **minimal** if $\nexists$ a DFA N s.t. $L(N) = L(M)$ & N has fewer states than M.

Ex M 

is minimal wrt Σ^*

Proof technique Proving that a machine M is minimal.

① Construct minimal DFA $M = (Q, \Sigma, \delta, q_0, F)$

↳ Each state should encode a not redundant/unique piece of information

(2) Argue M is minimal

(A) Consider arbitrary DFA N

(B) Find $|Q|$ strings using M

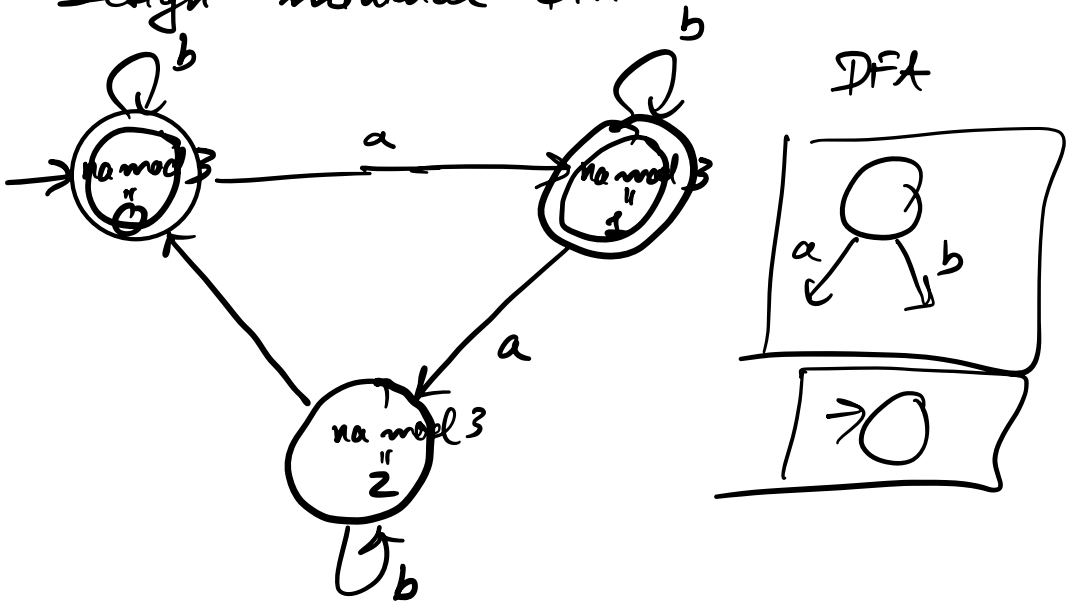
(C) Argue that each pair of strings must go to a different state in N

Ex Design a minimal DFA M s.t.

$L(M) = \{ w \in \{a,b\}^* : n_a(w) \bmod 3 = 0 \text{ or } 1 \}$

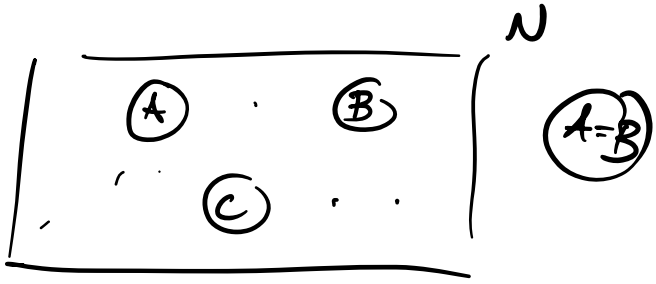
↳ # of a's in w

1. Design minimal DFA



2. (A) Consider arbitrary DFA N

s.t. $L(N) = L(M)$



(B) $w_0 = \varepsilon$ $w_1 = a$ $w_2 = aa$

Suppose w_0 goes to A,

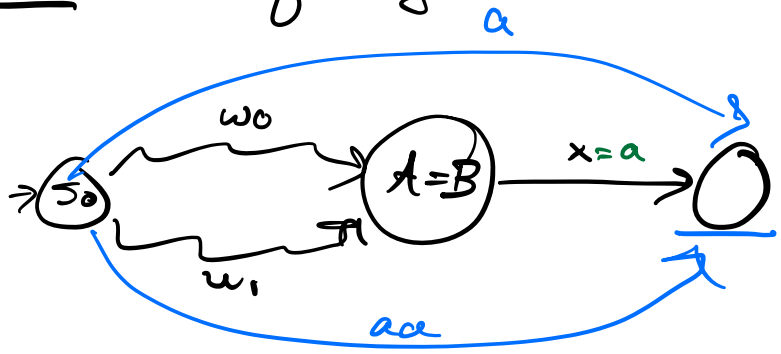
w_1 goes to B,

w_2 goes to C.

(C) Argue $A \neq B$, $B \neq C$, $C \neq A$.

$A \neq B$ Proof by contradiction.

N



$w_0 x$ & $w_1 x$ should go to the same state

Suppose $x = a$

$w_0 x = w_0 a = \varepsilon \cdot a = a$

$w_1 x = a \cdot a = aa$

This is impossible since $n_a(a) = 1 \bmod 3 = 1$ (should be accepted)

$n_a(aa) = 2 \bmod 3 = 2$ (should not be accepted)

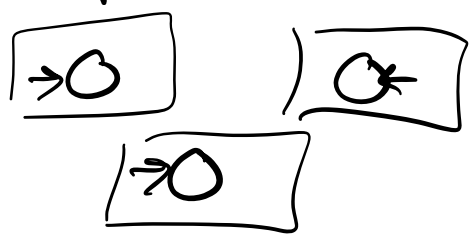
Repeat this for $B \neq C$, $C \neq A$.

Non-deterministic finite automata (NFA)

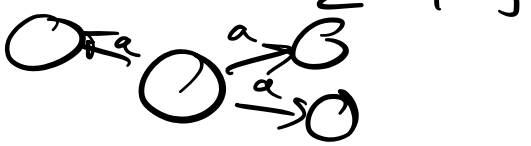
In DFA, the behaviour of the machine M is uniquely determined by the state it's in and the symbol it's reading. That is, DFA don't guess / make choices.
What happens when give an FA multiple possible choices.

Def (Informal) An NFA is a comp. machine which is similar to DFA except that it is allowed to have 0 or more "transition choices."

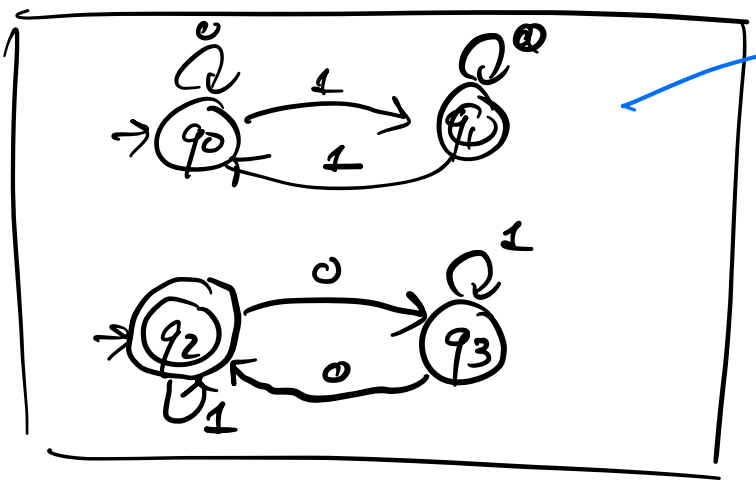
Multiple start states



Multiple transition choices $\Sigma = \{a, b\}$



Ex NFA N $\Sigma = \{0, 1\}$



DFA from Lecture 3

$L(N)$ =

$\{x \in \{0,1\}^* :$ $n_1(x)$ is odd OR $n_0(x)$ is even$\}$

Input: $x = 1\ 0\ 0\ 1$

$\to q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_1 \xrightarrow{0} q_1 \xrightarrow{1} (q_0)$, Failed computation

$\to q_2 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \to q_2 \xrightarrow{1} (q_2)$, Successful comp.

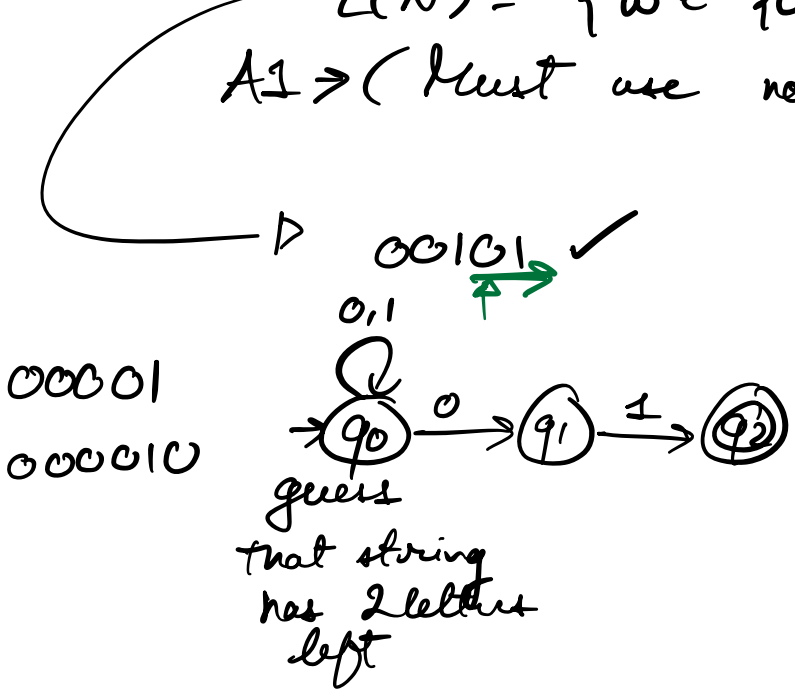
Since N has at least 1 successful comp., it accepts x

Input: $x = 1\ 0\ 1$ is rejected.

In general, N accepts a string x <=>
$\exists$ one successful comp: N starts at a start state, N reads the entire string x, N ends in an accept state.

Exercise Design a DFA that accepts $L(N)$.

Exercise Design an NFA N s.t.
$L(N) = \{w \in \{0,1\}^* :$ w ends in $01\}$
A1 -> (Must use non-determinism)



00101 ✓

00001

00010

$x = 0\ 0\ 1$

$q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0$ (Machine too patient)

$q_0 \xrightarrow{0} q_1$ -> Jams at q_1 Reject.

$q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_1 \xrightarrow{1} (q_2)$

success

$x = 001$ is accepted

$x = 0010$

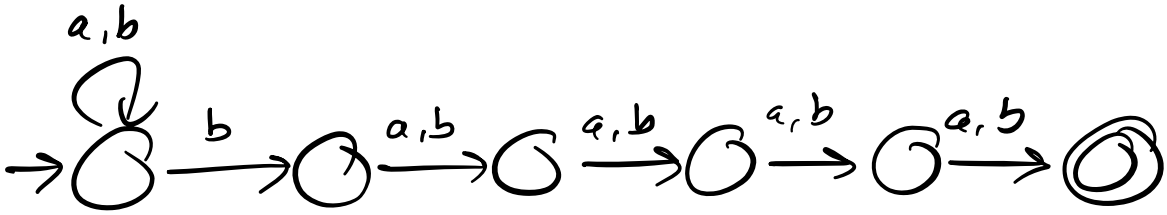
↑ ends at q_2

Not a success.

Exercise Design an NFA that accepts the language $\{w \in \{a,b\}^* : |w| \geq 5,$ the fifth to last letter of w is $b\}$ → $2^5 = 32$

$x = ab\,b\,aaab$

↑ 5th to last



Def (NFA) An NFA N is a 5-tuple $(Q, \Sigma, \Delta, S_0, F)$ Σ, Σ^*

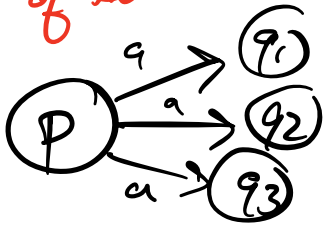
Q: finite set of states

Σ: input alphabet

Δ: transition function

$\Delta : Q \times \Sigma \to 2^Q$

The data type: a set of subsets of Q



S_0: set of start states $S_0 \subseteq Q$ $\{\Delta(p,a) = \{q_1, q_2, q_3\}$

F: set of accept states, $F \subseteq Q$