first words spoken → write algorithm / recipe to prepare coconut chutney based on DAA course done earlier (xD)

split:

* mid sem - 25
* end sem - 50
* assignment - 25

Text books →
① Class Notes
② Hopkraft & (theory) Ullman.
③ Peter Linz (problems)

purpose of studying TOC

(actually why study at all? go to ISCKON akshayapatra, get free food & clothes no?):

* bring discipline into thinking.
* convert intuition into logic so that the proof is universal.
* try to prove the smallest of axioms, even if intuitive.
* develop skillset. (most important, focus)
* convert thinking into systematic form (algorithm) → and make into a form that computer understands (computational thinking)

* give back something to society.
* what else? (Assignment 1)
* i/p (or) o/p doesn't matter process matters.
  ↳ if IK algo perfectly, why does the 1st chutney I make is worse than my mother's? (experience, intuition, skills)
* prove $2^0 = 1$, $0! = 1$.
* find height of obj using its image.

can my program work for ANY arbitrary input (?)
↳ is it countably infinite set of i/p s ?

* can we print n'th prime
* can we print ALL $\mathbb{R}$ no's in some (a, b) range?

* intel celeron floating pt error.
* smart attendance system.

Objective of TOC:

* limitations of computing
* prove formally - solvable / unsolvable
* model computations.

* find unsolvable problems in each course. (DSA, DLD, DAA, PSP, DSCS)

adding constraints ⟶ might lead a well known solvable problem into an unsolvable one
(sorting ⟶ constant time sorting)

Belief ⟶ NP-Hard problems don't have a poly-time algorithm.
We dK whether ⟶ $P \neq NP$ or $P = NP$

Given a well defined problem:

i/p :
? :
o/p :

⟶ does this have algorithm?
Yes → solvable
No → unsolvable

Studying this classification is theory of Computation ⟵ The science in Computer Science.

Compute models/machines for this:
(say calculator, computer, coffee machine, humans, etc...)

↳ need language for acheiving purpose
- humans → natural language
- machines → digital language (voltages) → +5V / 0V

THEORY of COMPUTATION / COMPUTING / COMPUTING SYSTEMS / AUTOMATON
requires:

models / machines	Language	Computing Power	Representation.

MACHINES → understand only pressing of appropriate buttons in a specific way (?) (action depends on machine)
All machines are same → if looked as a computing element

"can unsolvable become solvable"
"every machine has its limitations"
Keep asking questions until you hit a non-trivial one → make that your final year project → take time
($q_1 \to q_2 \to \cdots \to q_{opt}$, q_{opt} → non-trivial, LOL)
unsolvable → result of inherent complexity of problem + the limitations of the machine.

1) Finite State Automaton → Automate the manual process represent the computing part using states, & use finitely many such states.

theres an infinite tape, divided into cells - each cell → {0,1}

A cell → can store one of 0, 1 or $

the machine can:

↳ read cell by cell, move right one step at a time

↳ read symbol.

the machine can't:

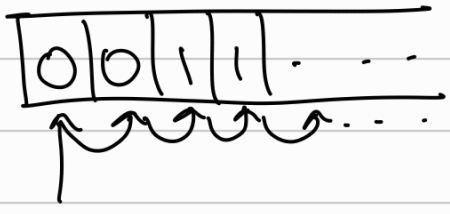
↳ jump cells

↳ read words

↳ write

↳ store words

<u>α-bet</u> : $\Sigma = \{0, 1\}$



READ & MOVE

i/p : $x \in \Sigma^*$; $\Sigma = \{0, 1\}$, Σ^* → set of all strings over Σ

? : does M accept x

$S \subseteq \Sigma^*$, $S = \{\phi, 01, 001, 101, 1101, 0101, \cdots\}$

can we think of FSA to recognise 'S'

i/p : x → $x \in S$ – yes

→ $x \notin S$

wrt $S \rightarrow$ strings ending with 01

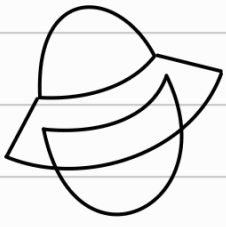
i/p: $S \subseteq \Sigma^*$ + any string in S ending with 01

design FSA $\rightarrow$ to recognise / accept S.

S = prefix + 01

= any prefix over $\{0,1\}$ + 01

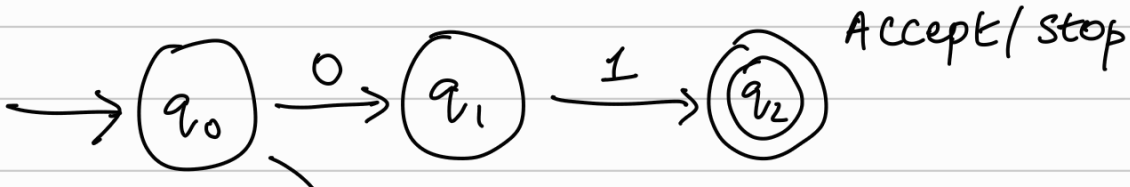
$\hookrightarrow$ check if belongs to Σ^* + 01



LOWER BOUND THEORY : min T.C. reqd. to solve. (like sorting - $n \log n$)
($\Omega(f(n))$)

α-bet $\rightarrow$ set of symbols.

somehow store the prefix in the FSA.
States - talk about intermediate computations



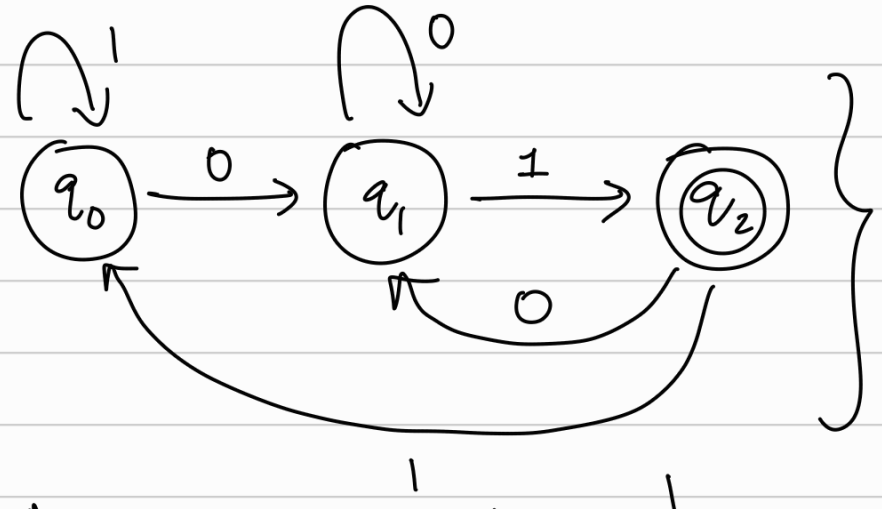
take all into acc

COULD

* read ∞-ly many 1's (say) before 01
* read ∞-ly many 0's before 01
* some arbitrary i/p before 01

* run a loop until all continuous 1's / 0's exhausted. Do this until in the END u come at 01.

↘ one for each State.



→ ACCEPTS if machine stops at q_2. else, NO.

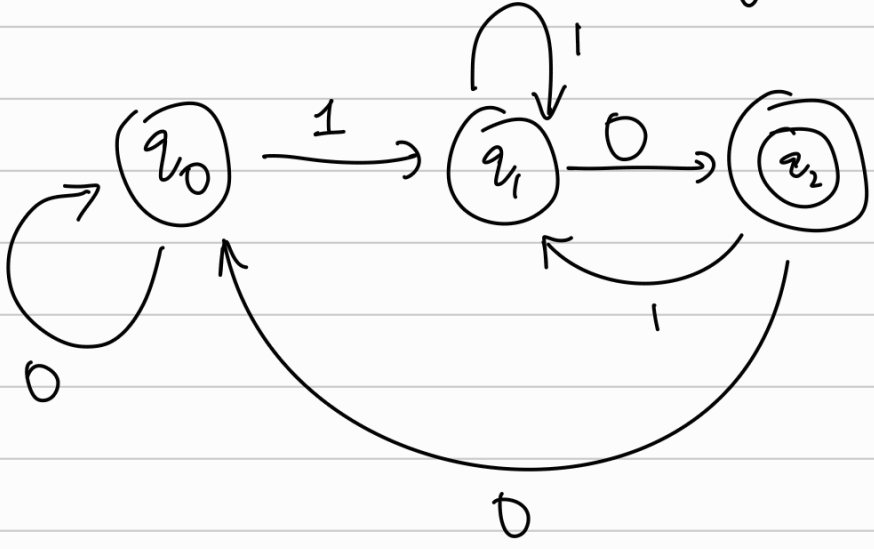
↑ TRY for all i/ps. ↓ FSA.

Simulate x on FSA. if FSA stops at (q_2), ACCEPT.
else, REJECT

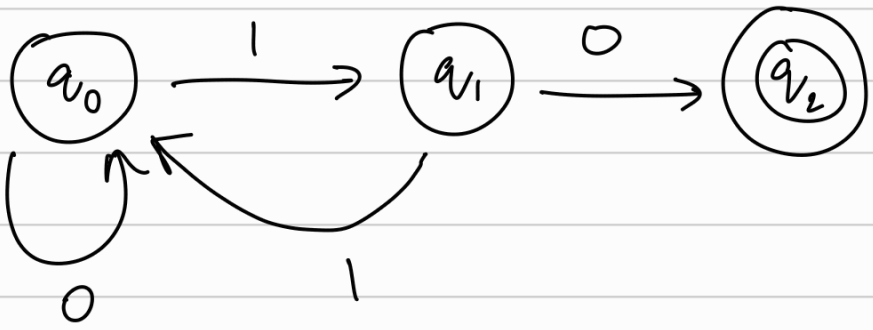
↗ Stopping at q_0 or q_1

a) Set of all strings ending with 10
b) set of all strings containing 10

a)



b)



$$FSA = (Q, \Sigma, \delta, q_0, F)$$

- Q → set of states
- Σ → i/p. α-bet.
- δ → transition fn.
- q_0 → start state
- F → final state

FSA → 5-tuple machine

δ : DOMAIN → CO-DOMAIN.

$$Q \times \Sigma \longrightarrow Q$$

$L = \{ x \mid x \text{ ends with } 01 \}$ $\quad \Sigma^* \rightarrow$ countably ∞.

$x \in \{0,1\}^* \rightarrow$ set of all strings over $\{0,1\}$

ε, 0, 1, 00, 01, 10, 11, and so on.....

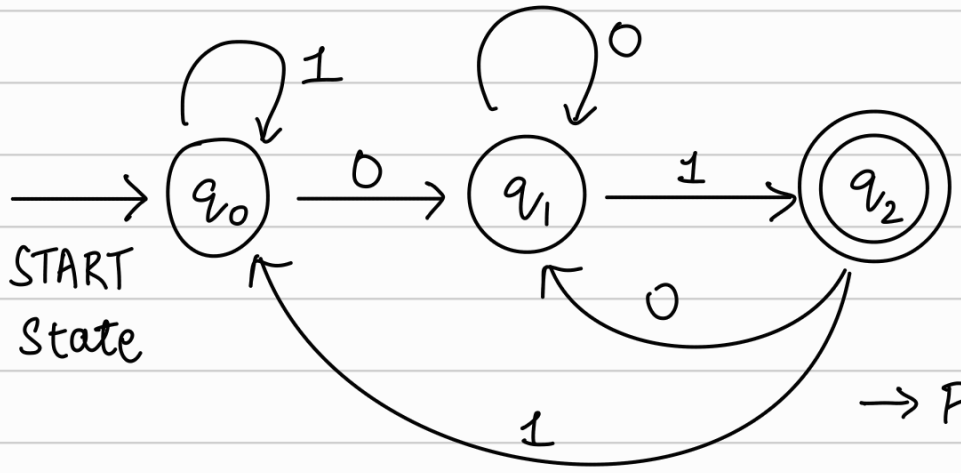
0 1 2

① can we design an FSA to accept Σ^*

② " " " " " " " $S \subseteq \Sigma^*$

$x \rightarrow \cdots 01.$

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
$\downarrow$
$\{q_0, q_1, q_2\}$



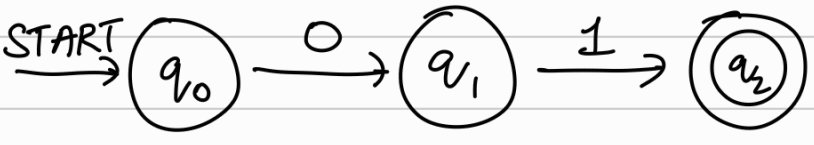
→ FINITE STATE DIAG

$\delta : (Q \times \Sigma) \rightarrow Q$ e.g. $(q_0, 1) \rightarrow q_0$
$(q_0, 0) \rightarrow q_1$

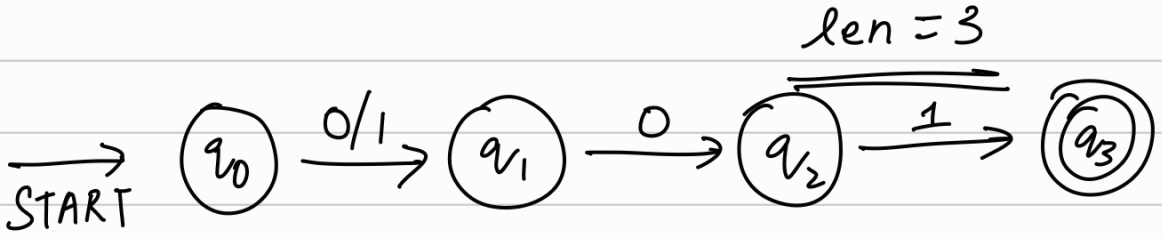
the state that i am in & the symbol i encounter decides the next state i have to be in → $Q \times \Sigma \rightarrow Q$

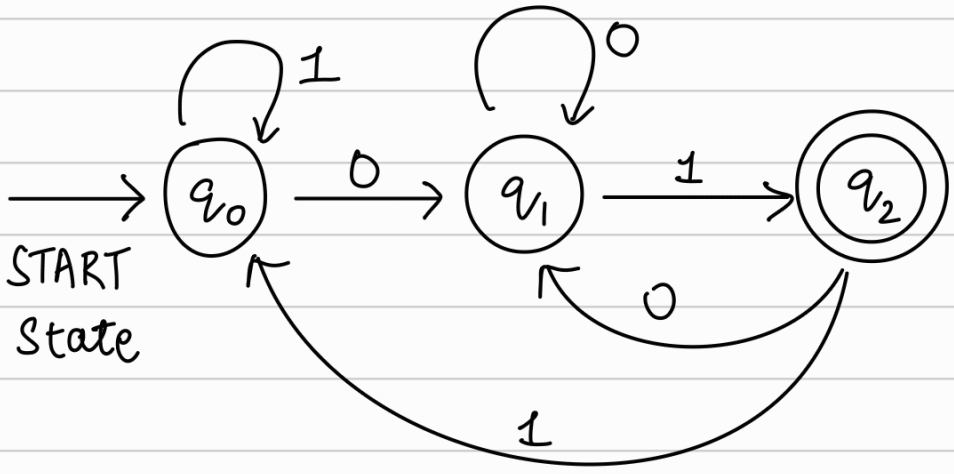
Steps to creating machine:
- ↳ just accept 01
- ↳ accept len = 3 + ends with 01
- ↳ " len = 4 + ends with 01
- ⋮ ⋮ ⋮ ⋮
- ↳ reject strings not ending with 01.

① 0-1 :



② 0-1 + len = 3





$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
↓
$\{q_0, q_1, q_2\}$

$(q_0, 0)$	$\rightarrow q_1$
$(q_0, 1)$	$\rightarrow q_0$
$(q_1, 0)$	$\rightarrow q_1$
$(q_1, 1)$	$\rightarrow q_2$
$(q_2, 0)$	$\rightarrow q_1$
$(q_2, 1)$	$\rightarrow q_0$

So, the diag respects fn. defn. . . .

FSA → MUST ACCEPT ALL VALID STRINGS (end with 01)

FSA → MUST REJECT ALL INVALID STRINGS (not ending with 01)

if $(q_2, 0)$ goes to q_0 instead of q_1, then if 0 in 01 is discovered, another 0 has to be discovered to verify i/p → WRONG logic !! → the machine stops at q_0, and not at q_2, if 0101

∴, $\delta(q_2, 0) = q_0$ is not right ! //

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

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

"unless I discover 01, I keep rejecting".
"I accept if I exist at q_2 (I have 01) and I have no more to discover".

if $\delta(q_2, 1) = q_1$, then 0111 will be accepted!
$\delta(q_2, 1) = q_2$, then 011 will be accepted!
$\delta(q_2, 1) = q_0$ → then things work....

∴, $\delta : (Q \times \Sigma) \to Q$ is well defined for the FSA....

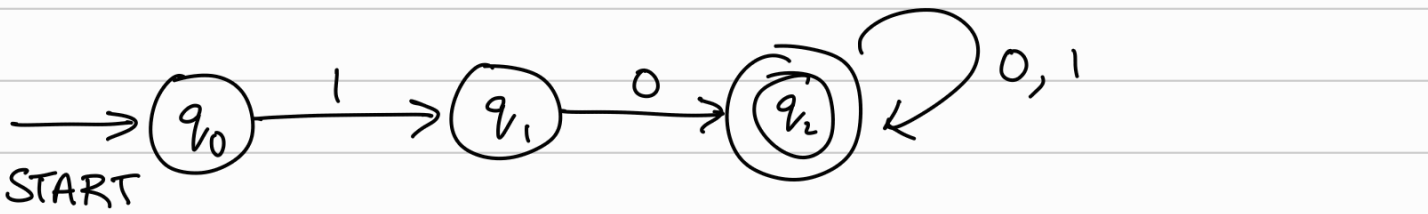
$$FSA = \{Q = \{q_0, q_1, q_2\}, \Sigma = \{0,1\}, \delta, q_0, F = \{q_2\}\}$$

Qs: Can we model a program for hostel lift using FSA???

$$\Sigma = \{0,1\}, \quad L = \{x \mid x \text{ begins with } 10\}$$

$\underbrace{10}_{l=2}, \underbrace{100, 101}_{l=3}, \underbrace{1001, 1010, 1011, 1000}_{l=4}$, & so on...

10 + suffix = $y \in \{0,1\}^*$



defined: $\delta(q_0, 1)$, $\delta(q_1, 0)$, $\delta(q_2, 1)$ and $\delta(q_2, 0)$. Others?

$$M = \{ Q = \{q_0, q_1, q_2\}, \Sigma = \{0, 1\}, \delta, q_0, F = \{q_2\} \}$$

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

is δ_M well defined?

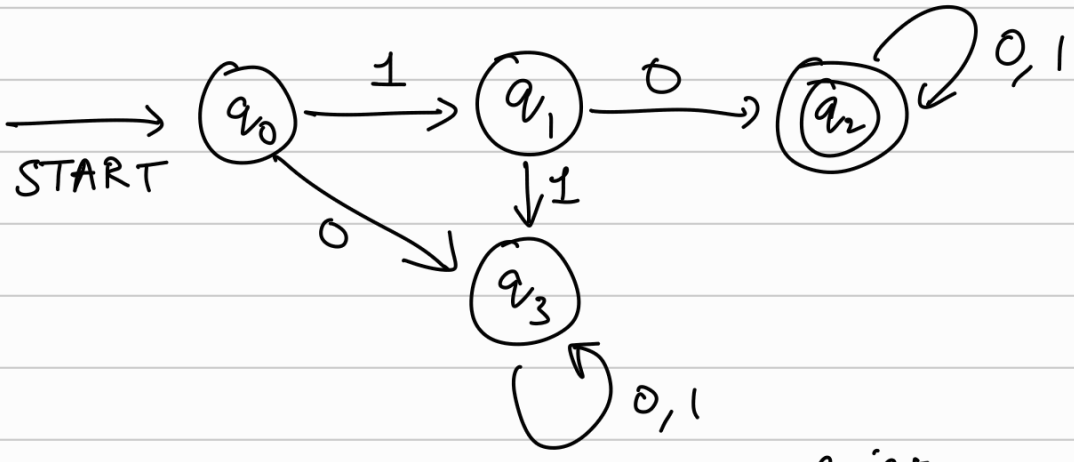
$(q_0, 0) \longrightarrow q_3$ INVALID
$(q_0, 1) \longrightarrow q_1$
$(q_1, 0) \longrightarrow q_2$
$(q_1, 1) \longrightarrow q_3$ INVALID
$(q_2, 0) \longrightarrow q_2$
$(q_2, 1) \longrightarrow q_2$

All strings $\in \{0,1\}^*$ beginning with 0 are invalid!

introduce an "invalid"/"reject" state – q_3

∴, updated M & diag:

$$M = \{ Q = \{q_0, q_1, q_2, q_3\}, \Sigma = \{0, 1\}, \delta, q_0, F = \{q_2\} \}$$



$\delta(q_3, 0) = q_3$
$\delta(q_3, 1) = q_3$
$\} \longrightarrow$

existence at q_3 itself means disagreement.
∴, exist there ∀ symbols after disagreement.

$\therefore$, $\delta_M : Q \times \Sigma \longrightarrow Q$ is well defined.

$(q_0, 0) \longrightarrow q_3$
$(q_0, 1) \longrightarrow q_1$
$(q_1, 0) \longrightarrow q_2$
$(q_1, 1) \longrightarrow q_3$
$(q_2, 0) \longrightarrow q_2$
$(q_2, 1) \longrightarrow q_2$
$(q_3, 0) \longrightarrow q_3$
$(q_3, 1) \longrightarrow q_3$

{ ending at ANY state other than q_2 is REJECTION... }

is the above a min-state automaton?
(min state FSA) ↳ Prove / Disprove....

<u>1-state FSA</u> :

(e.g). $M = \{ Q = \{q_0\}, \Sigma = \{0,1\}, \delta, q_0, F = \{q_0\} \}$

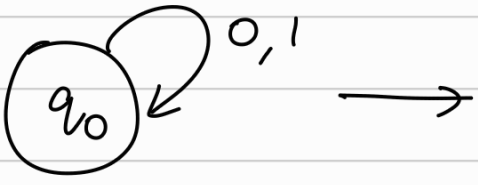
$\delta : \{q_0\} \times \{0,1\} \rightarrow \{q_0\}$

{ $F \subseteq Q$ and not $F \in Q$
↳ NEED NOT BE UNIQUE
$q_0 \in Q \rightarrow$ UNIQUE }

$\delta \rightarrow$ model the computation
program / algorithm / procedure

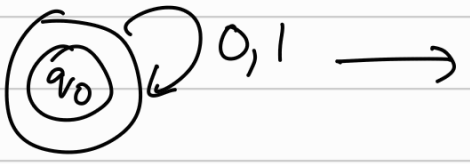
$F \rightarrow$ can be empty / contain multiple states.....
$\therefore$, subset of Q.

* fix $Q = \{q_0, q_1, q_2\}$
* identify language for each final state (?)
  ↳ $n(\text{languages}) = n(\text{subsets}) = 2^n$.
  $n \rightarrow$ no. of final
* problem is solved → if language identified for each states (?)



$F = \emptyset$, $L = \emptyset$ } → accepts nothing....

FSA



→ FSA accepts everything...

$L \subseteq \{\varepsilon, 0, 1, 00, \ldots\}$

$L \in \text{PowerSet}(\Sigma^*)$

} didn't understand language part well. hope to understand as course progresses.

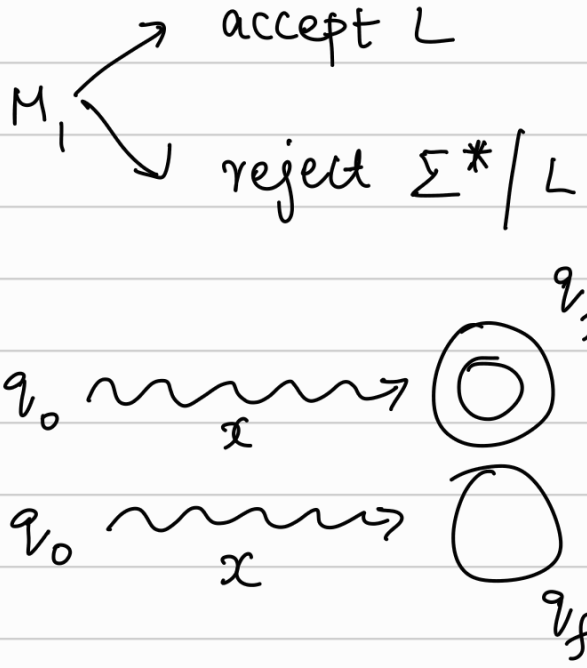
30th Jan -

$$L_1 = \{x \in \{0,1\}^* \mid x \text{ ends with } 01\}$$

$$L_2 = \{x \in \{0,1\}^* \mid x \text{ starts with } 10\}$$

language → set of all strings accepted by M.

$$L(M_1) = \Sigma^*$$



$L(M) \longrightarrow$ language accepted by FSA M.

$\hookrightarrow L(M) = \{ x \mid \delta(q_0, x) = q_f \in F \}$

essentially, $L(M) \subseteq \Sigma^*$

① given an arbitrary subset of Σ^*, does ∃ a FSA "M" accepting S ?

$\delta(q_0, x) \longrightarrow$ take e.g :

$$\delta(q_0, 01) = \delta(\delta(q_0, 0), 1)$$

$$\delta(q_0, x) = \delta(\delta(\cdots(\delta(\delta(q_{f-1}, 0), 1), \cdots))$$

↓ saying ending with 01

δ: $\delta(q, a) = q' \in Q$.
$a \in \Sigma$

$\hat{\delta}$: $\hat{\delta}(q, x)$

$x \in \Sigma^*$ $\qquad x = x_1 x_2 x_3 \cdots$

$$\hat{\delta}(q_0, x_1 x_2 x_3 \cdots)$$

$$= \hat{\delta}(\delta(q_0, x_1), x_2 x_3 \cdots) = \hat{\delta}(q_1, x_2 x_3 \cdots)$$

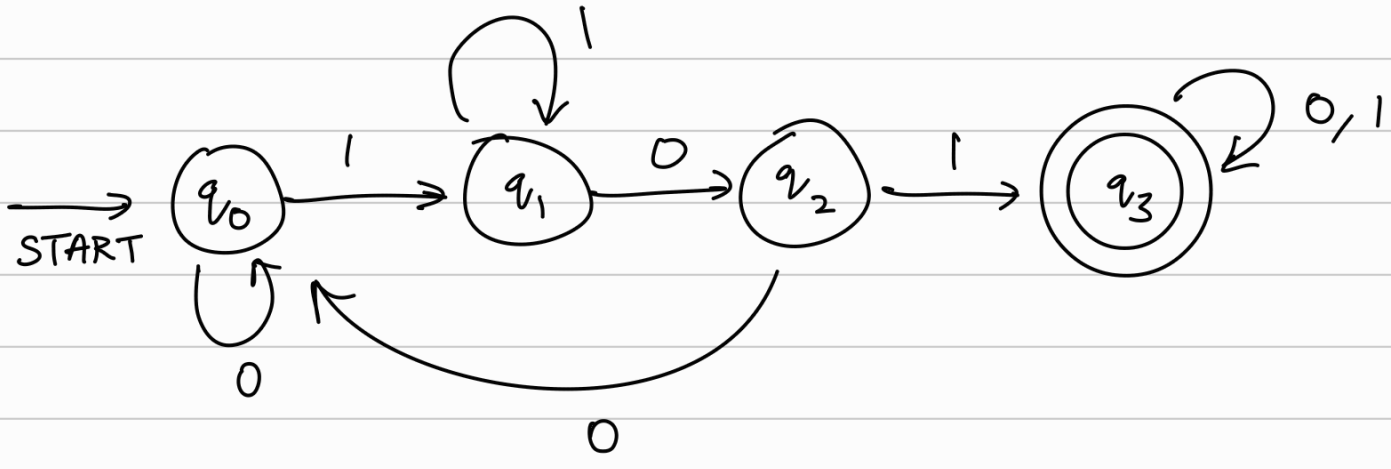
$$= \hat{\delta}(\delta(q_1, x_2), x_3 x_4 \cdots) = \hat{\delta}(q_2, x_3 x_4 \cdots)$$

and so on, until:

$$= \hat{\delta}(\delta(q_{f-1}, x_f), \text{NULL}) = \hat{\delta}(q_f, \text{NULL}) = q_f$$

↳ $\$$

$$L_3 = \{ x \in \{0,1\}^* \mid x \text{ contains substring } 101 \}$$



$$M = \{Q = \{q_0, q_1, q_2, q_3\}, \Sigma = \{0, 1\}, \delta, q_0, F = \{q_3\}\}$$

$$\delta : Q \times \Sigma \longrightarrow Q$$

$\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_2$

$\delta(q_1, 1) = q_1$ → AND NOT q_0, as 1101 won't be accepted!

sequence doesn't break too. so, no prob...

$\delta(q_2, 0) = q_0$ → AS the sequence breaks, so must search for 101 from first 101 discovered.

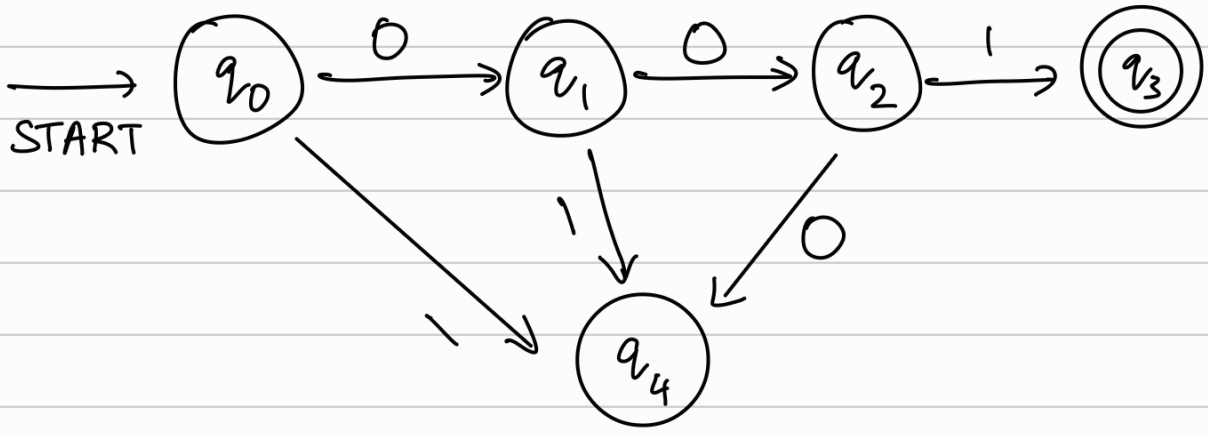
$\delta(q_2, 1) = q_3$ →

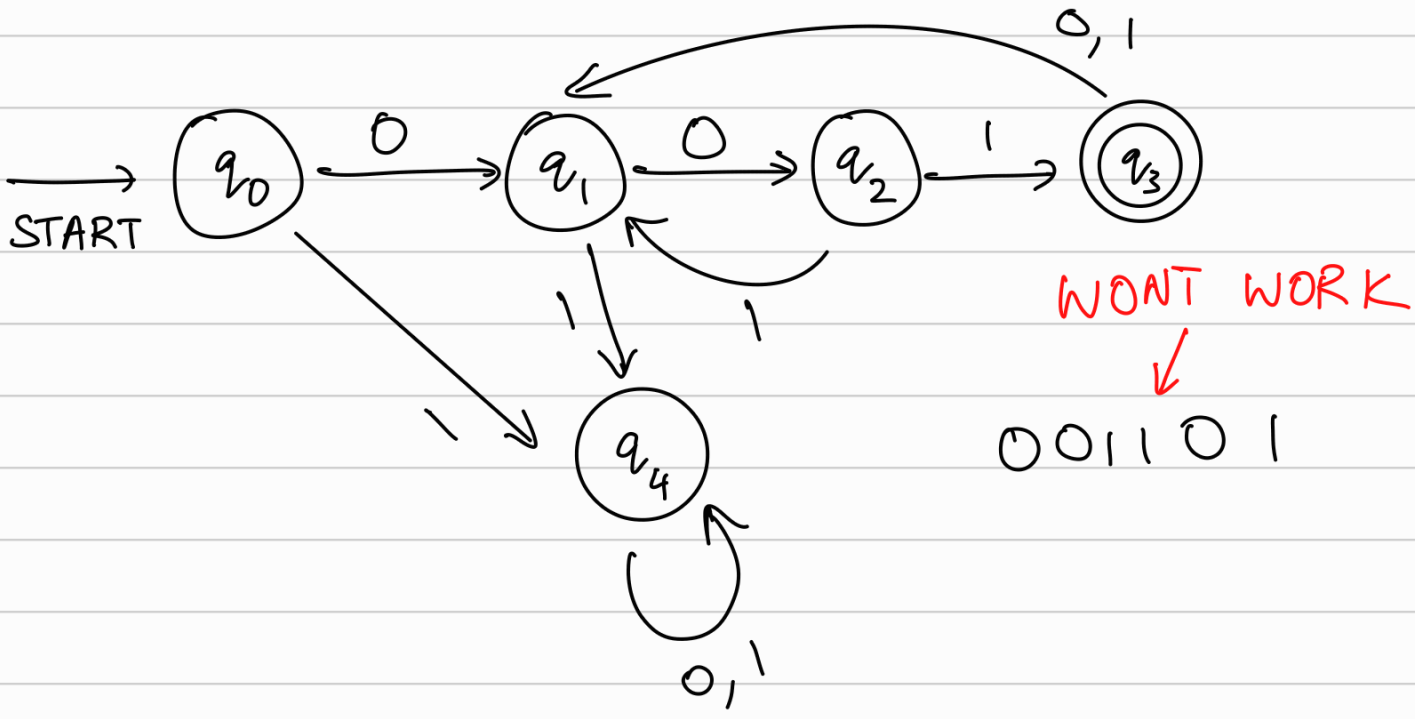
$\delta(q_3, 0) = q_3$ } → STAY at q_3 once 101 is discovered, happily :)

$\delta(q_3, 1) = q_3$ }

$$L_4 = \{x \in \{0,1\}^* \mid x \text{ begins with } 00 \text{ \& } x \text{ ends with } 01\}$$

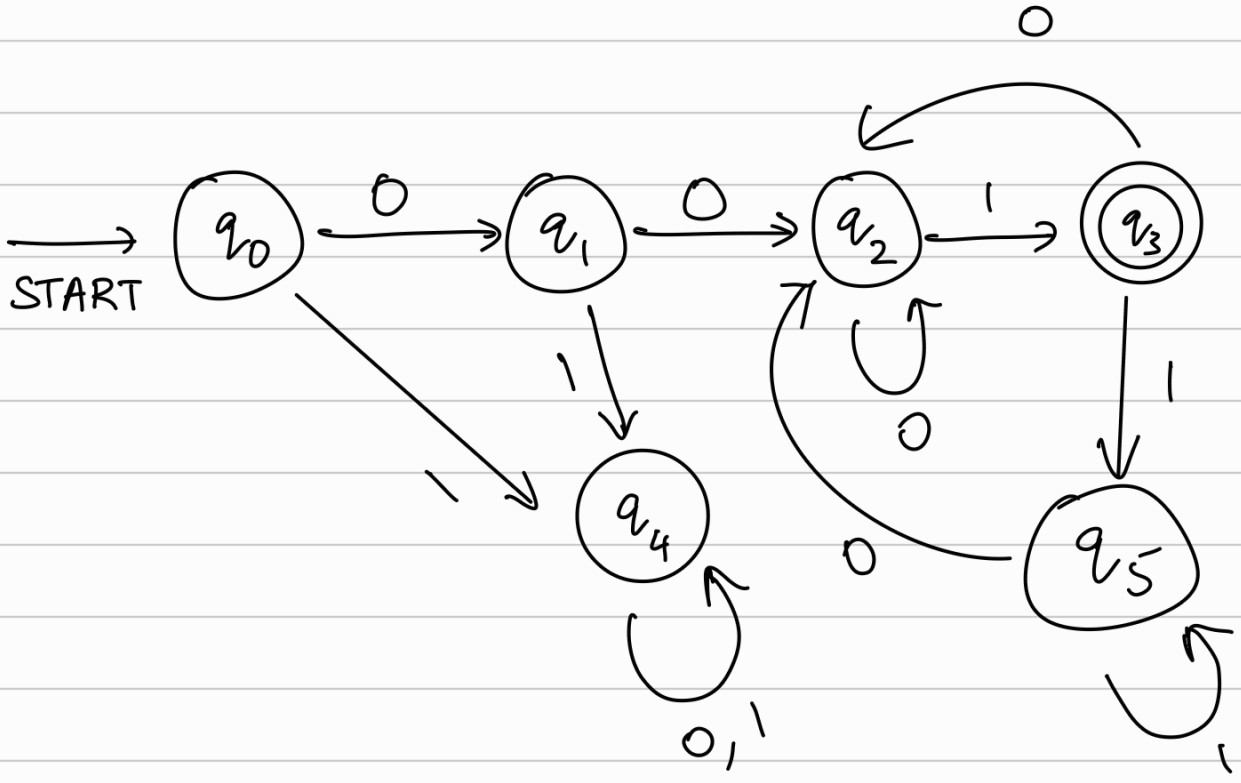
try with 001 (& then finetune...)





WONT WORK

0 0 1 1 0 1



WHAT MISTAKES occurred :

↳ Self looping at q_4 at 0 0 1 1 0 1 . . . .

↳ shdn't go back to any of the prev states . . .

∴, introduce q_5 .

$\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_4$ (REJECT)

$\delta(q_1, 0) = q_2$

$\delta(q_1, 1) = q_4$ (REJECT)

$\delta(q_2, 1) = q_3$

$\delta(q_2, 0) = q_2$ (self-loop, in case many 0's occur - preserve sequence)

$\delta(q_3, 0) = q_2$ (to check if 01s happen again - q_2 preserves sequence anyways)

$\delta(q_3, 1) = q_5$ (in case sequence breaks, restart at q_5)

$\delta(q_5, 0) = q_2$ (sequence started, so go to q_2 & try to finish/preserve)

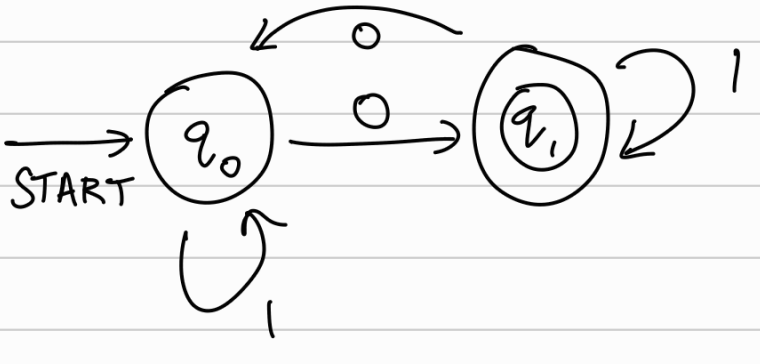
$\delta(q_5, 1) = q_5$ (sequence didn't start, wait until it does).

$$L_4 = \{ x \in \{0,1\}^* \mid x \text{ has even 1's \& odd 0's} \}$$

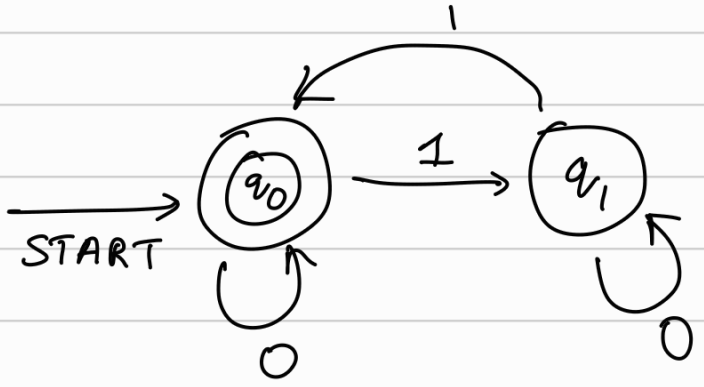
Observations:

* x is ALWAYS of ODD length....
* self-loop DOESN'T aid in counting.

Q1) Odd no. of 0's → no constraint on 1's.



Q2) even no. of 1's → no constraints on 0's.



Q3) How to unify both of these ???

L_4 → essentially odd #0s ∧ even #1s.

$= Q_1 \wedge Q_2 \cdots$

questions

$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow \cdots \rightarrow q_n$

"Keep Asking Questions" – SADDY :)

↓ GREEDY

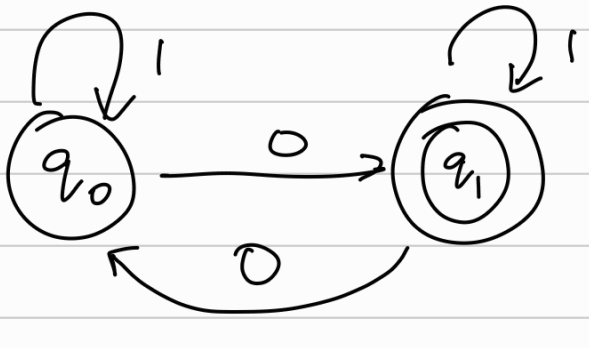
q_n → extraordinary

X ——— X ——— X

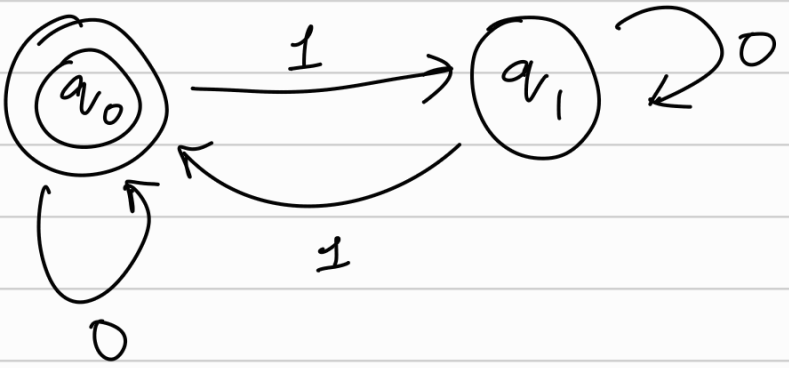
$L_1 = \{ x \in \{0,1\}^* \mid x \text{ has odd no. of 0's} \}$

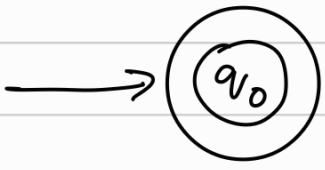
$L_2 = \{ x \in \{0,1\}^* \mid x \text{ has even no. of 1's} \}$

M_1 :



M_2:

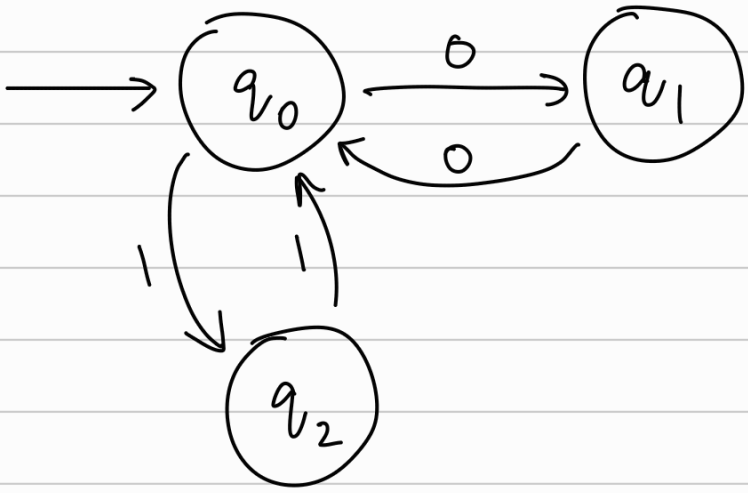




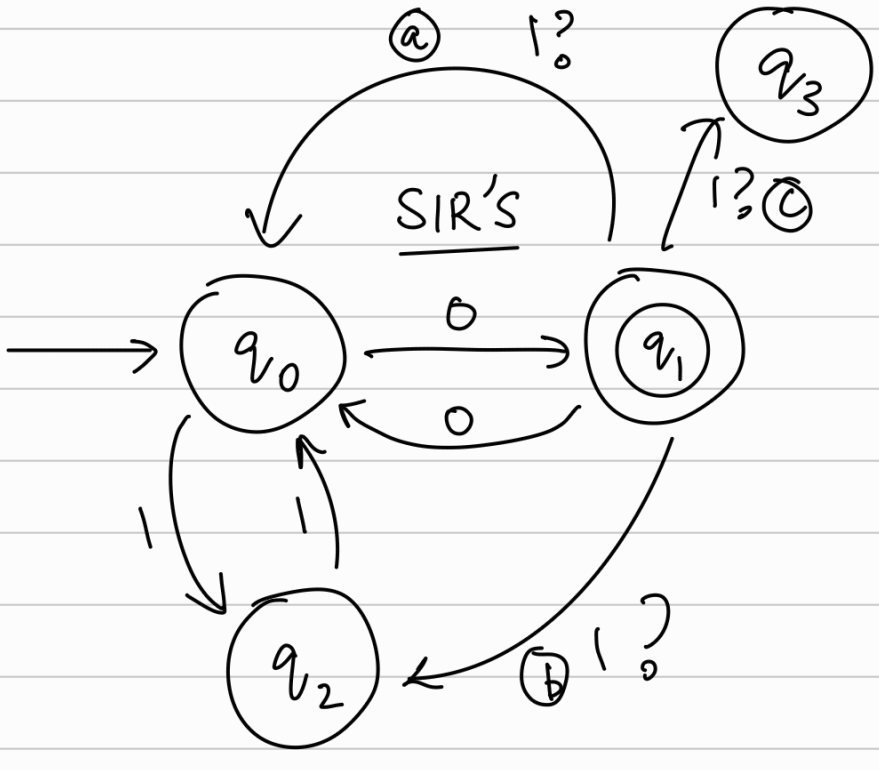
: FSA accepts a string without reading anything.
Accepts empty string – $\in$ (epsilon)
. in case of even no. of 1s.....

$$L_5 = \{ x \in \{0,1\}^* \mid x \text{ has odd no. 0's } \wedge \text{ even 1's} \}$$

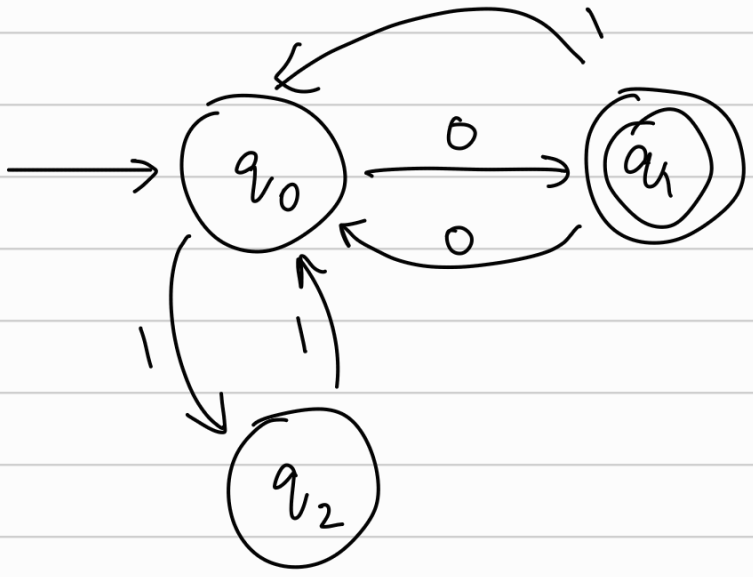
{ ∵ odd no of 0's, 1 is the least size of string. }



$\delta(q_0, 0) = q_1$
$\delta(q_0, 1) = q_2$
$\delta(q_1, 0) = q_0$
$\delta(q_1, 1) =$

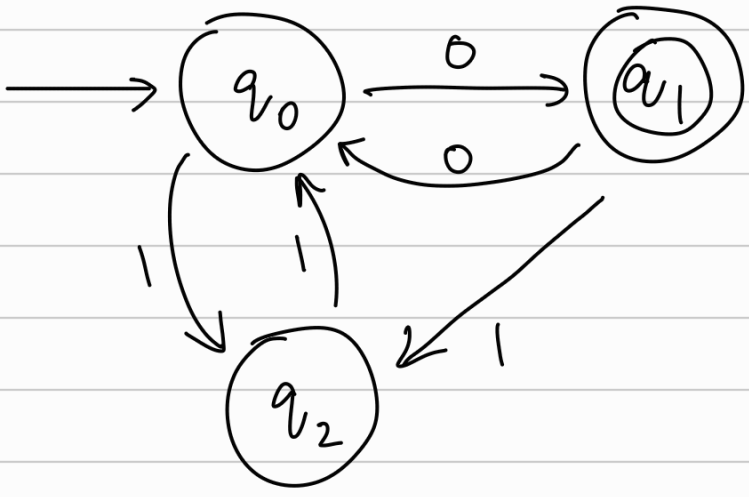


(a) let $\delta(q_1, 1) = q_0$



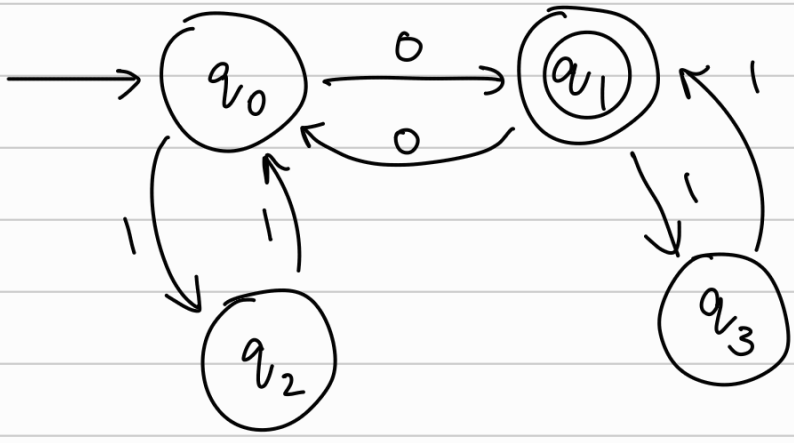
Machine accepts
010
↳ invalid.

(b) let $\delta(q_1, 1) = q_2$



Machine accepts
0110 → invalid.

(c) let $\delta(q_1, 1) = q_3$



let $\delta(q_3, 1) = q_1$.
∴, the loop
$q_1 - q_3 - q_1$
tracks even 1's
like $q_0 - q_1 - q_2$

What about $\delta(q_3, 0)$?

a) $q_0 \rightarrow$ 0100 is accepted (incorrect) - odd 1's
b) $q_1 \rightarrow$ 010 is accepted (incorrect) - even 0's
c) $q_2 \rightarrow$ seem's right ....

$\therefore$, $\boxed{\delta(q_3, 0) = q_2}$ //

01010
00110111010
accepts
0111010

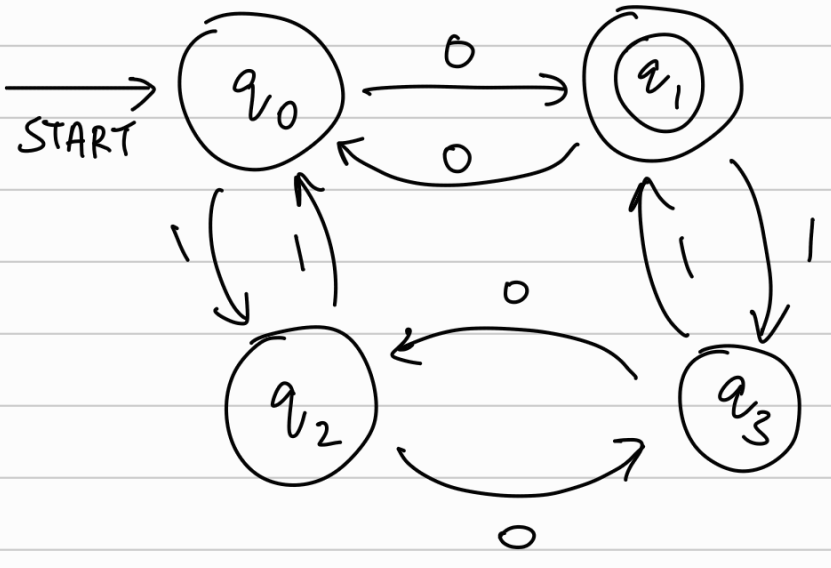
What abt $\delta(q_2, 0)$?

a) q_0 - 100 is accepted
b) q_1 - 10 is accepted
c) q_2 - self loop, hence wrong
d) q_3 - seems right..

$\therefore$, $\delta(q_3, 0) = q_2$.

looking at $\delta: Q \times \Sigma \rightarrow Q$, 001110

$\delta(q_0, 0) = q_1$ -
$\delta(q_0, 1) = q_2$ --
$\delta(q_1, 0) = q_0$.
$\delta(q_1, 1) = q_3$ ---
$\delta(q_2, 0) = q_3$ ---
$\delta(q_2, 1) = q_0$.
$\delta(q_3, 0) = q_2$ --
$\delta(q_3, 1) = q_1$ -



110010 001101

01, 10, 0001, 1110,
0111, 1000, 110010,
001101

we have seen counting & pattern matching problems.
is that all? what are the limitations of FSA???

gain intuition.

for the same automaton discussed,
what's the property of:

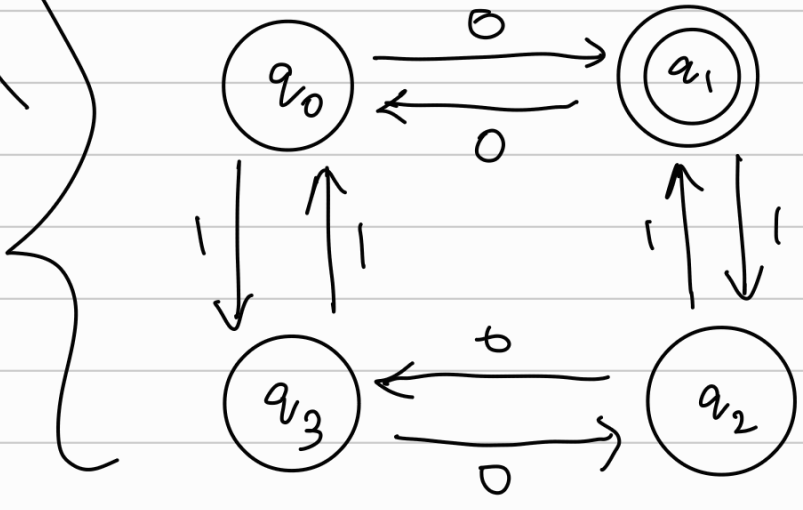
(i) $\hat{\delta}(q_0, x) = q_0$
(ii) $\hat{\delta}(q_0, x) = q_1$
(iii) $\hat{\delta}(q_0, x) = q_2$
(iv) $\hat{\delta}(q_0, x) = q_3$

MY OBSERVATIONS -

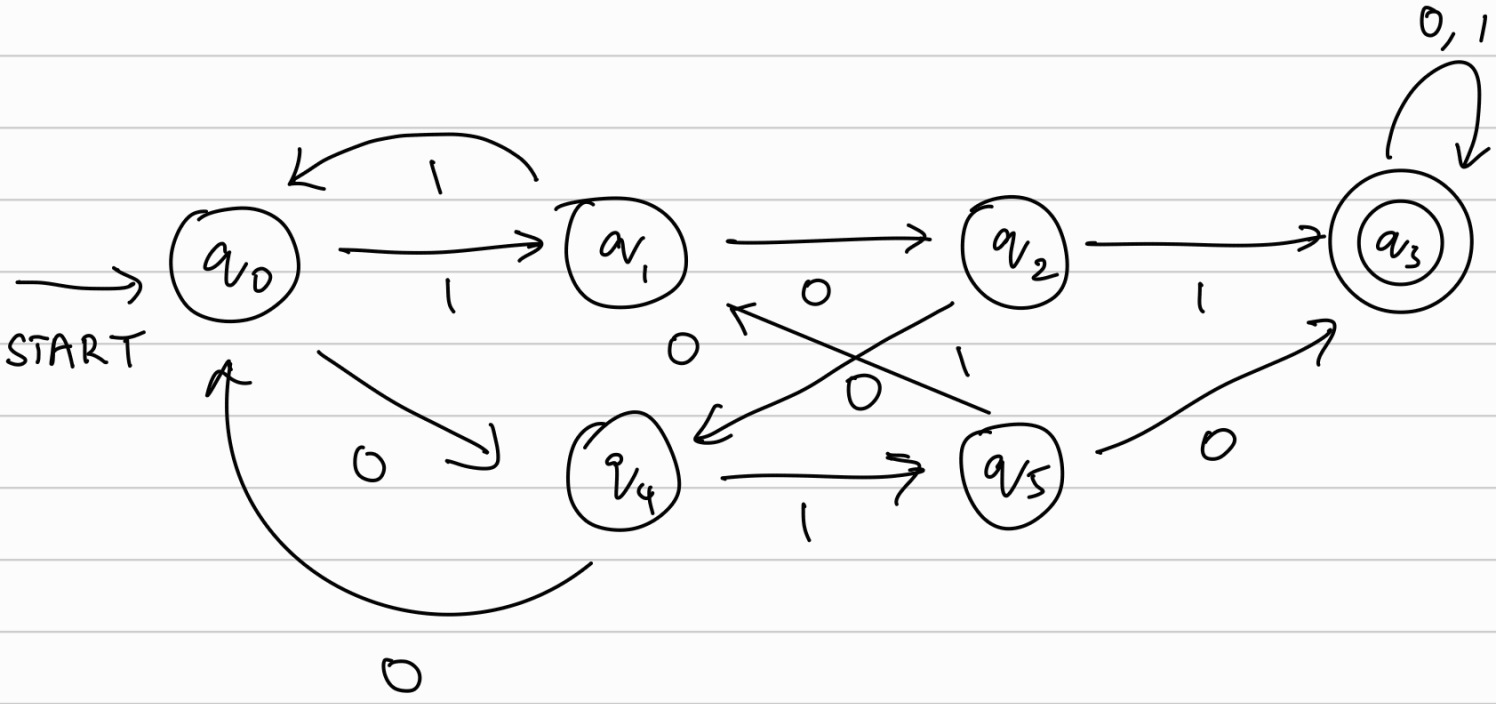
(i) even no. of 0's & 1's.
(ii) odd no. of 0's & even 1's
(iii) odd no. of 1's & even 0's.
(iv) * strings & their complements
* strings have even size.

Observe same for this.

the machine changed?



$L = \{ x \in \{0,1\}^* \mid x \text{ contains } 101 \text{ or } 010 \text{ as substring} \}$

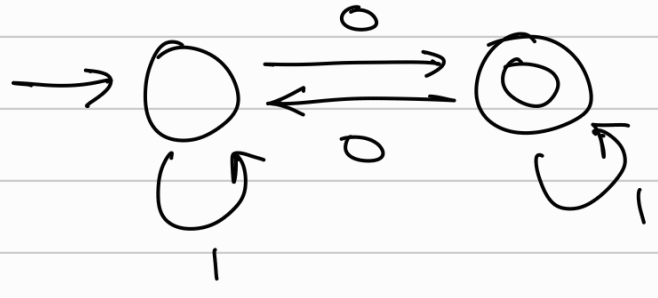


Given two FSAs M_1, M_2

$M_1 = \{Q, \Sigma, \delta_1, q_0^1, F_1\}$

$M_2 = \{Q, \Sigma, \delta_2, q_0^2, F_2\}$

M_1 :

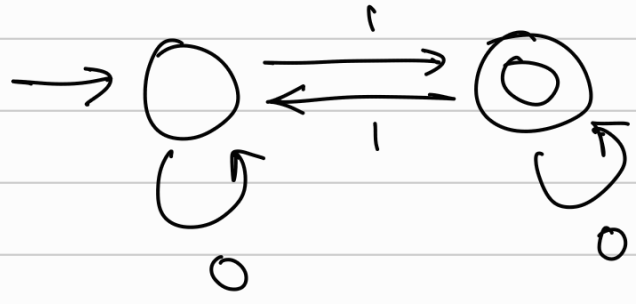


$M_1 \vee M_2$

(or)

$M_1 \wedge M_2$

M_1 :



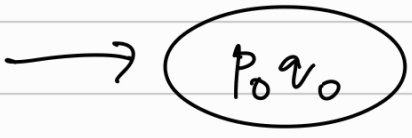
PRODUCT AUTOMATON:

$$M_1 \times M_2 = \{ Q_1 \times Q_2 , \Sigma , \delta , q_0 , F \}$$

$q_0 \rightarrow g(q_0^1, q_0^2)$ $\quad$ $F \rightarrow f(F_1, F_2)$

$$q_0 \in Q_1 \times Q_2$$

$$F \subseteq Q_1 \times Q_2$$



$$\delta((P_0, q_0), 0) = \delta(P_0, 0) \cup \delta(q_0, 0)$$

$$(Q_1 \times Q_2) \times \Sigma \rightarrow (Q_1 \times Q_2)$$

$$Q_1 = \{ q_0, q_1 \} \; ; \; Q_2 = \{ P_0, P_1 \}$$

$$Q_1 \times Q_2 = \{ (q_0, P_0), (q_0, P_1), (q_1, P_0), (q_1, P_1) \}$$

$$\delta((P_0, q_0), 1) = \left(\delta_2(P_0, 1) , \delta_1(P_0, 0) \right) = (P_1, q_0)$$

$$\delta((P_0, q_1), 0) = (\underbrace{\delta_2(P_0, 0)}_{\text{state of } Q_2}, \underbrace{\delta_1(q_1, 0)}_{\text{state of } Q_1}) = (P_0, q_0)$$

$$M_1 \times M_2 = \{ Q_1 \times Q_2, \Sigma, \delta, (q_0, P_0), F \}$$

to define $F \longrightarrow M_1 \wedge M_2$ ①
$\longrightarrow M_1 \vee M_2$ ②
$\longrightarrow M_1 - M_2$ ③
$\longrightarrow M_2 - M_1$ ④

① $\left\{ x \;\middle|\; \begin{array}{l} \hat{\delta}_1(q_0, x) = q \in F_1 \\ \hat{\delta}_2(P_0, x) = q' \in F_2 \end{array} \right\}$

$$\{ x \mid \hat{\delta}((q_0, P_0), x) = (q, q') \in F_1 \times F_2 \}$$

$\therefore M_1 \wedge M_2 = \{ Q_1 \times Q_2, \Sigma, \delta, (P_0, q_0), F_1 \times F_2 \}$

② $L = \left\{ x \;\middle|\; \begin{array}{l} \hat{\delta}_1(q_0, x) = q \in F_1 \\ \text{or} \\ \hat{\delta}_2(P_0, x) = q' \in F_2 \end{array} \right\}$

$$= \left\{ x \;\middle|\; \hat{\delta}((P_0, q_0), x) = (q, q'), \; \begin{array}{c} q \in F_1 \\ \vee \\ q' \in F_2 \end{array} \right\}$$

$$= \{ x \mid \hat{\delta}((p_0, q_0), x) = (q, q') \in (F_1 \times Q_2) \cup (Q_1 \times F_2) \}$$

captures all of $q \in F_1 \wedge q_1 \in F_2$, $q \in F_1$ & $q' \in Q_2$ & $q \in Q_1$ & $q' \in F_2$.

$$\boxed{(q, q') \in (F_1 \times Q_2) \cup (Q_1 \times F_2)}$$

Using the above model, construct 010 & 101 qs!

M_1 = 010 as substring
M_2 = 101 as substring.

} & then do $M_1 \cup M_2$ HOMEWORK.

given M_1, M_2 : how to construct M_1^c & M_2^c.

M_1^c: x doesn't contain y

M_2^c: x doesn't contain y'

$x \in L(M_1) \wedge x \in L(M_2) \Rightarrow x \in L(M_1 \wedge M_2)$

$$\delta((q_0^1, q_0^2), x) = \left(\delta(q_0^1, x), \delta(q_0^2, x)\right)$$

$q_f^1 \in F_1$, $q_f^2 \in F_2$. $\quad (q_f^1, q_f^2)$

$\hookrightarrow x \in L(M_1 \wedge M_2)$

if $x = (q_{nf}^1, q_f^2)$ (or) (q_f^1, q_{nf}^2) (or)

$(q_f^1, q_f^2) \longrightarrow x \in L(M_1 \vee M_2)$

if $x = (q_f^1, q_{nf}^2) \longrightarrow x \in L(M_1 / M_2)$

$x = (q_{nf}^1, q_f^2) \rightarrow x \in L(M_2 / M_1)$

$$M_1 \times M_2 = (Q_1 \times Q_2, \Sigma, \delta, (q_0^1, q_0^2), F)$$

$$\delta((q, q'), a) = (\delta^1(q, a), \delta^2(q', a))$$

$\hookrightarrow a \in \Sigma$

$\hat{\delta}((q_0^1, q_0^2), x) = \underline{\quad ? \quad} \longrightarrow$ based on machine...

$\hookrightarrow x \in \Sigma^*$

$F(M_1 \wedge M_2) \rightarrow M_1 \wedge M_2 \rightarrow F = F_1 \times F_2$

(both machines accept)

$F(M_1 \vee M_2) \rightarrow M_1 \vee M_2 \rightarrow F = (F_1 \times Q_2) \vee (Q_1 \times F_2)$

(either of them are true)

$F(M_1 \setminus M_2) \rightarrow (q_f^1, q_{nf}^2) \hookrightarrow F = F_1 \times F_2^c$

$= (F_1 \times (Q_2 \setminus F_2))$

↳ non-final states.

$F(M_2 \setminus M_1) \rightarrow F = F_1^c \times F_2$

$= ((Q_1 \setminus F_1) \times F_2)$

(e.g.) $\hat{\delta}((q_0^1, q_0^2), x) = (q, q') \in F_1^c \times F_2$

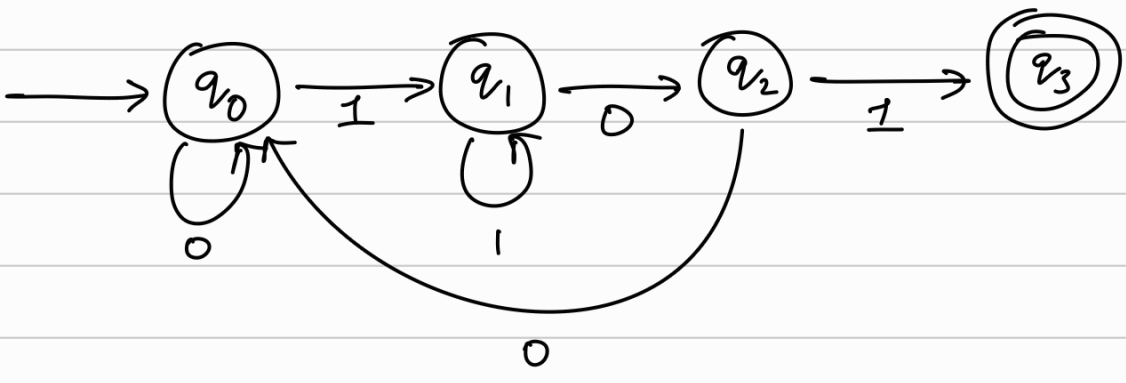
$\Rightarrow x \in L(M_2 \setminus M_1)$.

$M_1^c = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$. $F(M_1^c) = Q_1 \setminus F_1$

$L_1 = \{x \mid x \text{ doesn't contain 101 as substring}\}$

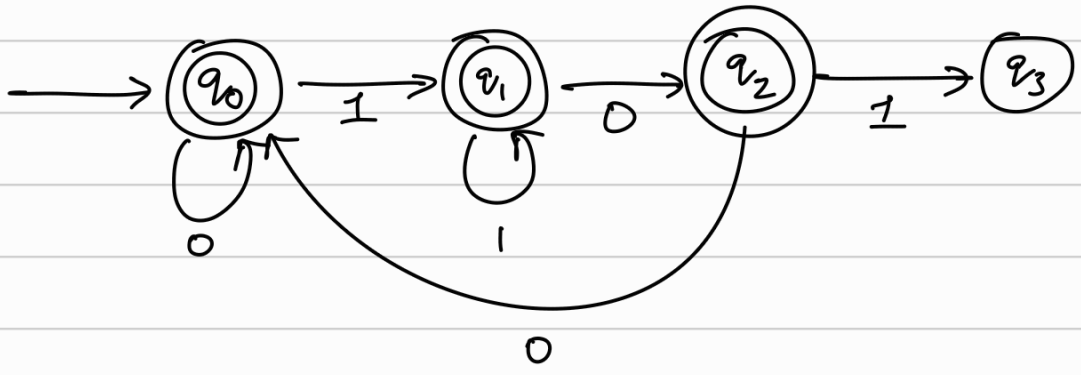
$M \equiv L = \{x \mid x \text{ contains 101 as substring}\}$

M :



$L(M_1) = L(M^c)$

M_1 (doesn't contain 101) :



"accept whats not by M, reject whats accepted by M"

given a Q → break down → L_1, L_2, L_3, ..., L_k → ans = $f_n(L_1, L_2, \cdots, L_k)$

(e.g.) $L = \{ x \mid x \rightarrow$ begins with 00 $\rightarrow L_1$, ends with 11 $\rightarrow L_2$, not containing 1010 $\rightarrow L_3^c \}$

$L = L_1 \wedge L_2 \wedge L_3^c$

$L_2 = \{ x \mid x$ contains $\underbrace{101}_{P_1}$ & doesn't contain $\underbrace{11}_{P_2} \}$

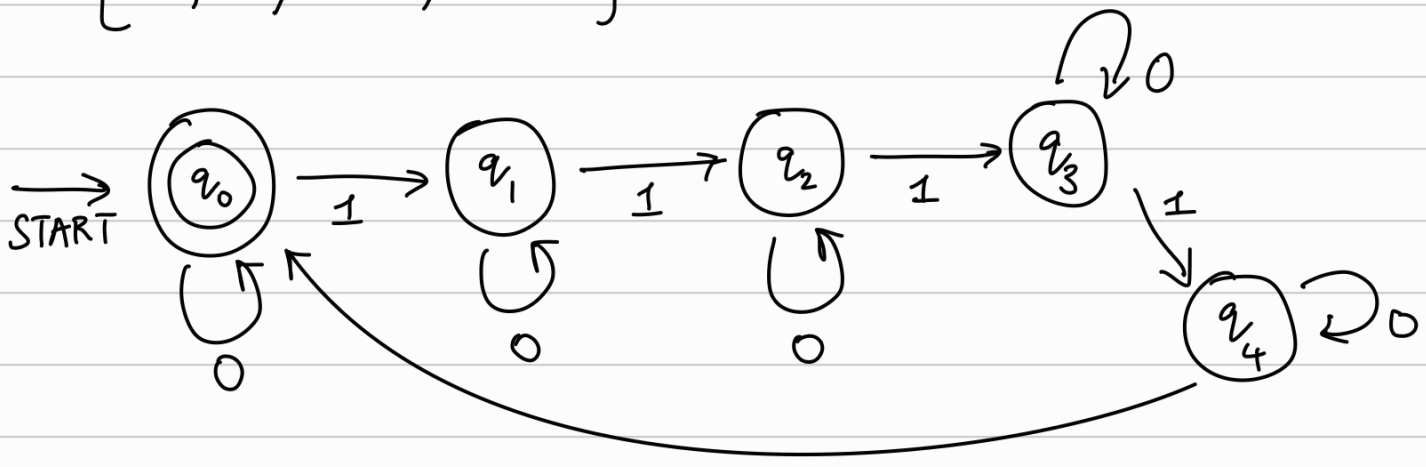
HOMEWORK.

P.T.O.

$$L = \{ x \mid \text{no. of 1's in } x \text{ is divisible by } 5 \}$$

$$L = \{ 0, 00, 000, \cdots, 0 \cdots 0, 0110111, \cdots\cdots, \}$$

$$L^c = \{ 01, 011, 0110, \cdots \}$$



each state keeps track of divisibility by 5. every 5th 1, the machine shall return back to $q_0 \longrightarrow$ accept. if !f div by 5 no. of 1's, we shall not end at q_0, hence $\longrightarrow$ reject.

can δ map $Q \times \Sigma$ to any subset of Q?

usually $\delta : Q \times \Sigma \longrightarrow Q$, meaning that an arbitrary ordered pair of q_i, a_i is mapped exactly to 1 next state. Such machines are "Deterministic Finite Automaton".

if this is the case, we dk. where the machine will reach for next state. such machines are "Non-Deterministic Finite Automaton".

Finite Automatons (FA)

- Deterministic FA (DFA): $\delta : Q \times \Sigma \to Q$
- Non-Deterministic FA (NFA): $\delta : Q \times \Sigma \to 2^Q$
  - 2^Q: Set of all subsets of Q (powerset)
  - indeed a fn., as each (q_i, a_i) is EXACTLY mapped to one subset S in 2^Q.

By observation,

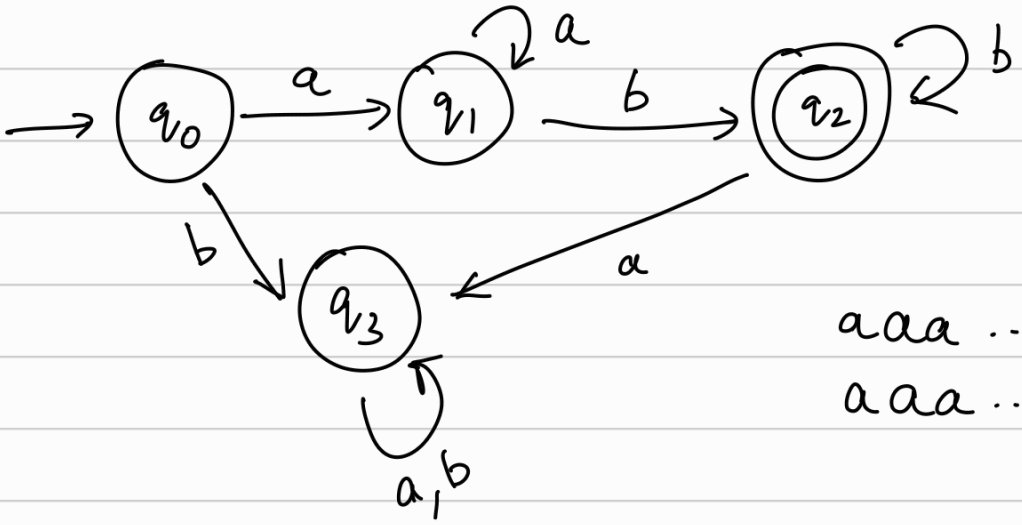
DFA is a special case of NFA → NFA is the generalisation that some q_i, a_i can be mapped to $\neq 1$ states (hence subsets of Q, hence $\delta : Q \times \Sigma \to 2^Q$)

QS: does NFA increase the computing power of the FA?

P.T.O.

$M = \{Q = \{q_0, q_1, q_2, q_3\}, \Sigma = \{a, b\}, \delta, q_0, F = \{q_2\}\}$

$\delta \rightarrow Q \times \Sigma \rightarrow Q$



accept.

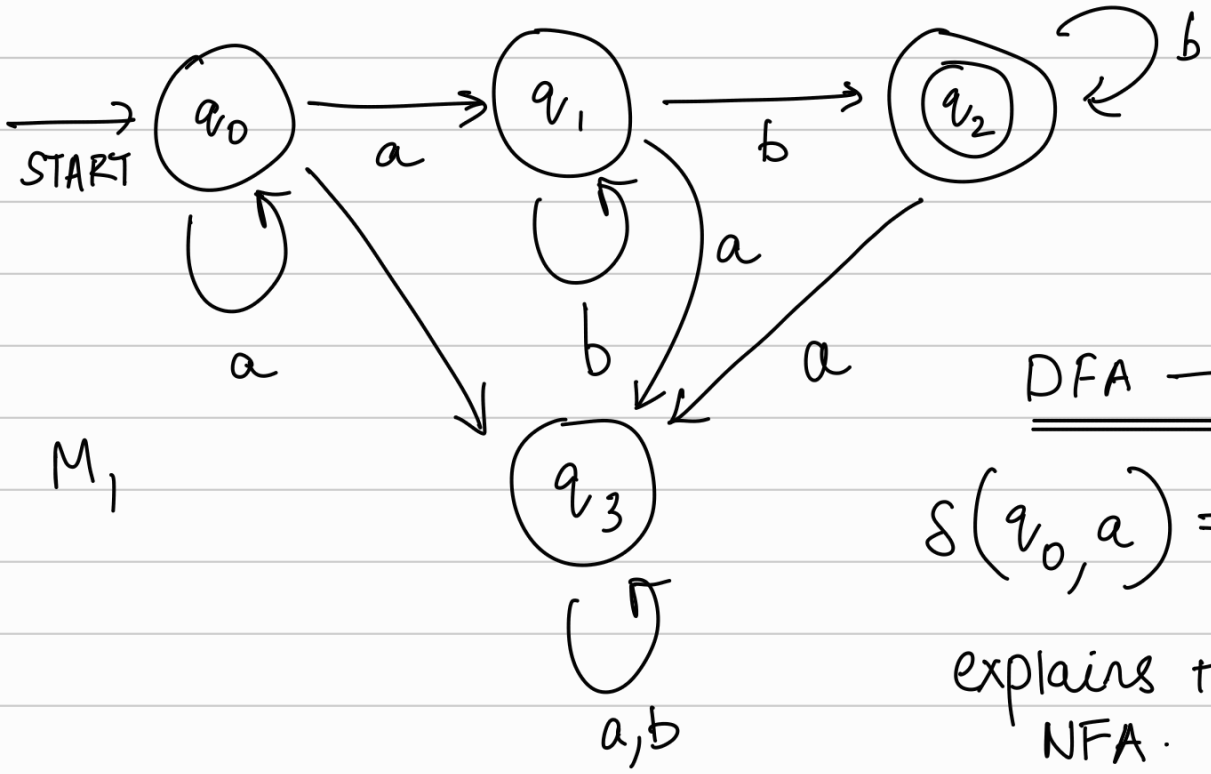
aaa .. abbb .. bb

aaa .. aabbb .. ba

reject

* reject beginning with b
* reject reading "a" after reading sequences of aa..aa b...b.
* chain initiation of a happens first, & initiation of a,b can happen exactly once...

language accepted $L = \{a^n b^m \mid n, m \geq 1\}$



M_1

DFA → NFA ?

$\delta(q_0, a) = \{q_0, q_1\}$

explains that its NFA.

$M_1 \longrightarrow$ accepts same language as M.

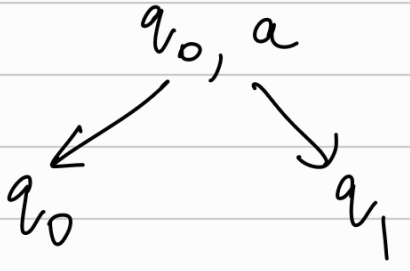
(e.g.) a^3b^2

→ $q_0 q_0 q_1 q_1 q_2$ shall work.

→ $q_0 q_0 q_1 q_2 q_2$ shall also work.

* my observation → choose next state as per ur discretion... the NFA provides a set of all possible "explorations" while parsing a string, & if ∃ one/more successful explorations, the machine accepts.....
(relate with depth first search??)

Qs: Is it true that for every NFA ∃ an equivalent DFA ?



for NFA ↴ the computational graph is a tree ↴ branch brings uncertainty, which brings non-determinism

for DFA → the graph is a path.

Class P → ∃ det. poly time algo ⟶ ∃ DFA
Class NP → ∃ non-det poly-time algo ⟶ ∃ NFA.

P = NP ⟶ if ∀ NFA ∃ DFA & vice-versa.

is non-determinism ACTUALLY non-determinism if machine does computations & then picks the best next state? is it so powerful that the guessing is quick? or does it perform magic?

DFA

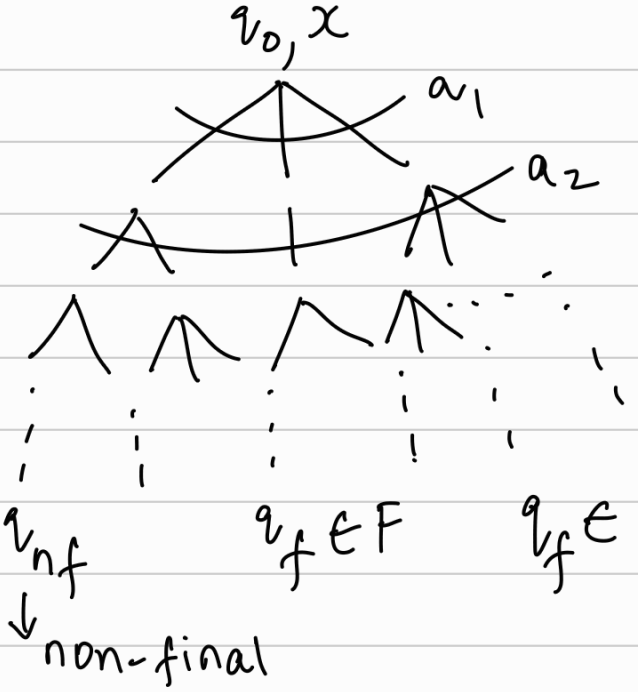
$$\delta(q_0, a) = q' \in Q$$

$$\hat{\delta}(q_0, x = a_1 a_2 \cdots a_n) = \underbrace{\left(\left(\left(\left((q_0, a_1), q_2\right), q_3\right), q_4 \cdots\right), a_n\right)}$$

$a_i \in \Sigma$

↳ ∃ a unique path.

$$L(M) = \left\{ x \mid \hat{\delta}(q_0, x) = q_f, q_f \in F \right\}$$



if ∃ one path in which $\hat{\delta}(q_0, x) = q_f \in F$, then x is accepted by the NFA. $x \in L(M)$
else, $x \notin L(M)$

if $\hat{\delta}(q_0, x) = q_{nf}$ for each path in tree, then $x \notin L(M)$.

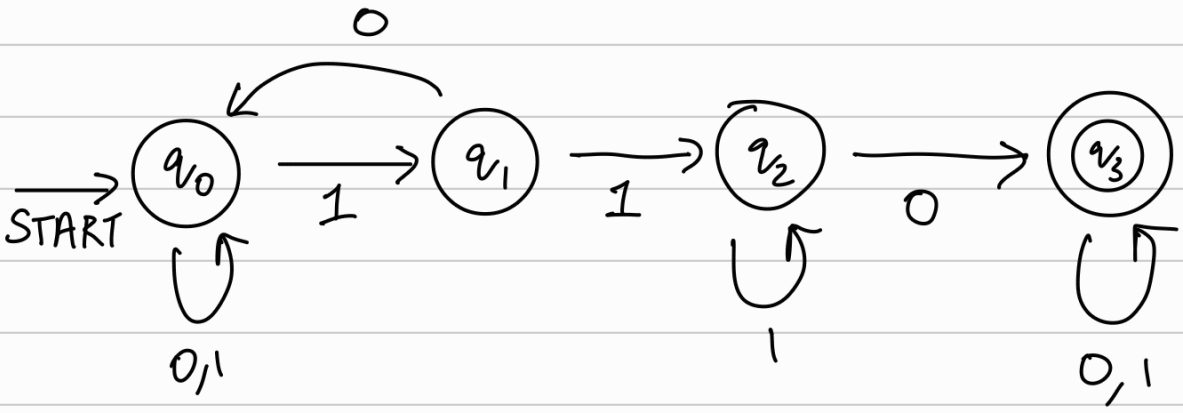
How do we discover the path? → guess
→ det. approach.

NFA → has power of guessing (guess can't be wrong)

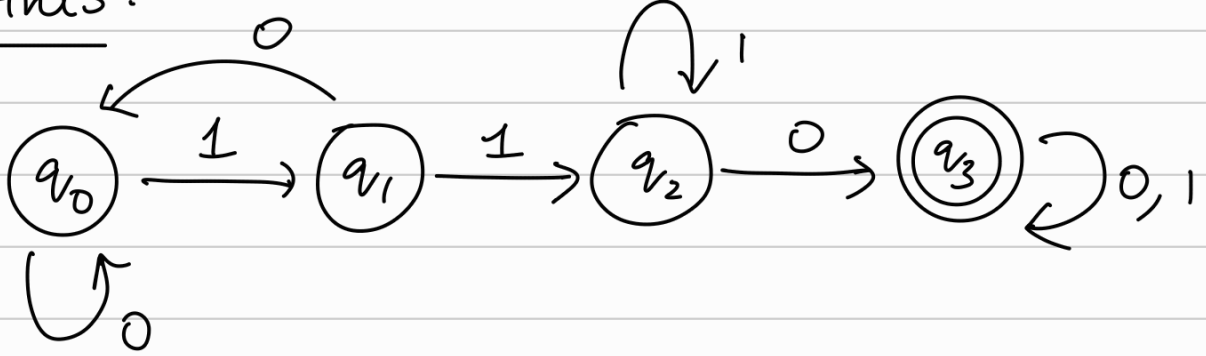
- if $x \in L(M)$, then NFA will INDEED stop at q_f, SOMEHOW discovers path from q_0 to q_f ....
- hypothetical only...
- can simultaneously simulate x @ q_0 (or) parallelly explore ALL paths...
- not AI as AI has inaccuracies....
- does introducing NFA increase the computing power of the automaton (solve something that a DFA can't?)

(e.g.) $L = \{ x \mid x \text{ contains } 110 \text{ as substring} \}$

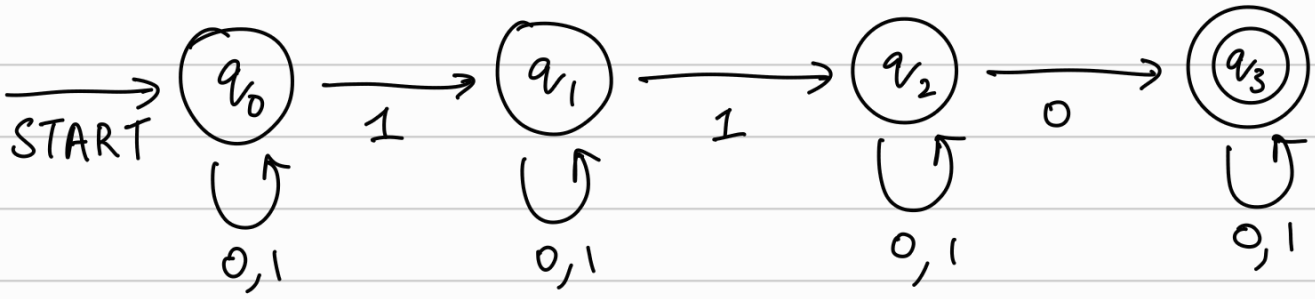
NFA for this: $\forall x \in L(M)$ $\exists$ path $q_0 \dots q_3$



DFA for this:



Another NFA proposed :



but! consider $x = 1010$. ∃ path $q_0 q_1 q_2 q_3$
AND! x doesn't have 110 substring, still machine accepts x. Hence, wrong!

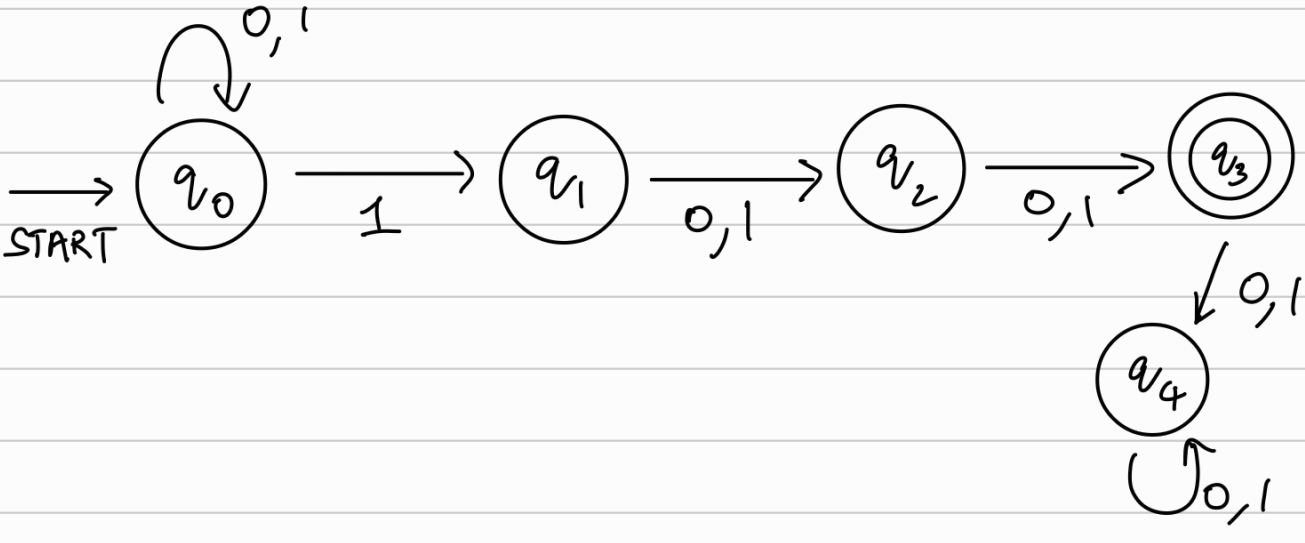
for FA: if valid str ⟶ ∃ path $q_0 \dots q_f$ //
invalid str ⟶ ∄ path $q_0 \dots q_{nf}$ //

$$L = \{ x \in \{0,1\}^* \mid \text{the third symbol from right is '1'} \}$$

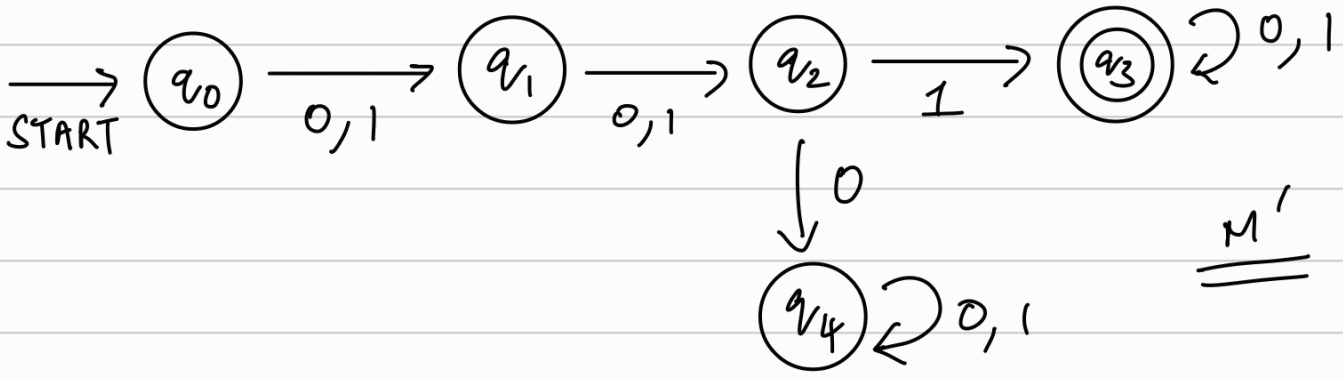
$$L = \{101, 000101, 10100110, 000100, \dots\}$$

$$L^c = \{000, 011, 10111011, \dots\}$$

does ∃ an FA for this ?



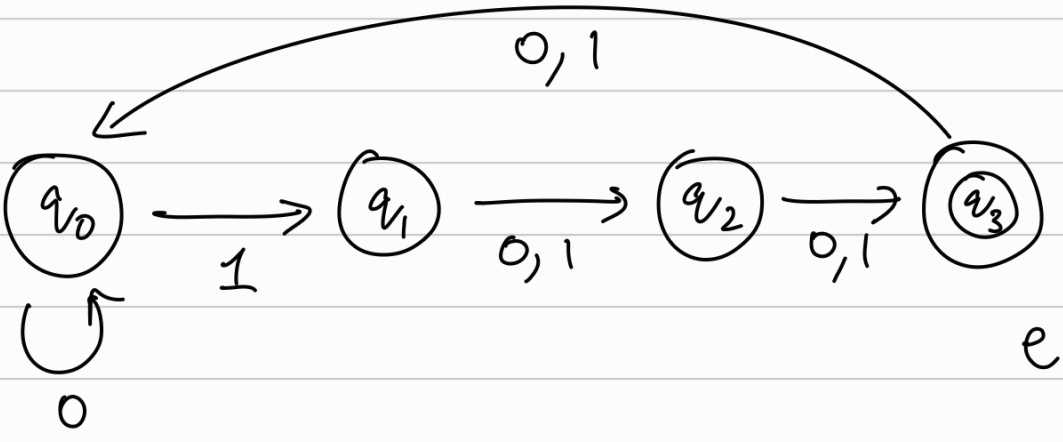
what if we reverse the string & find 3rd from left?



M'

can I use M' to acheive same goal as original machine ??

Also,

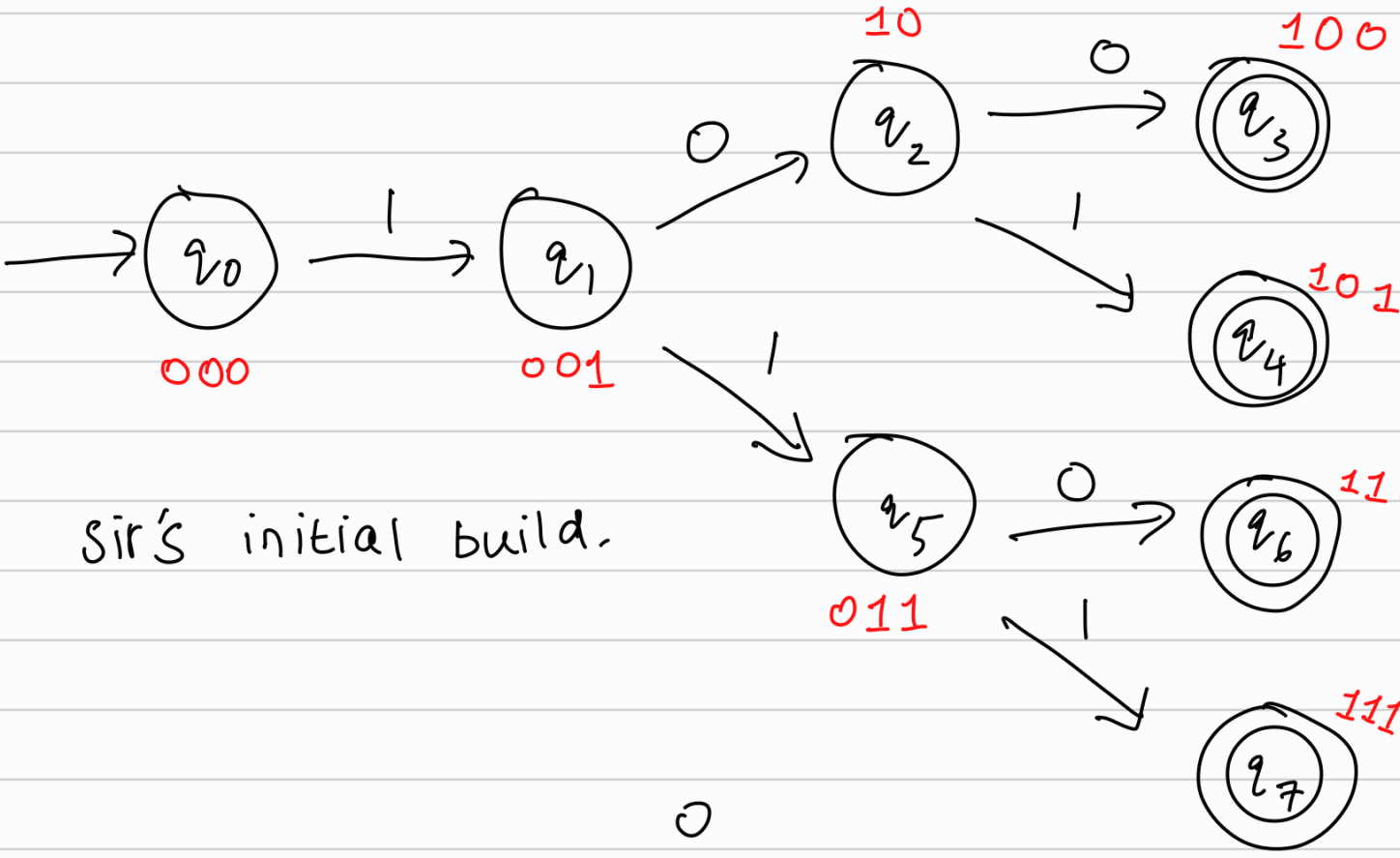


exists...

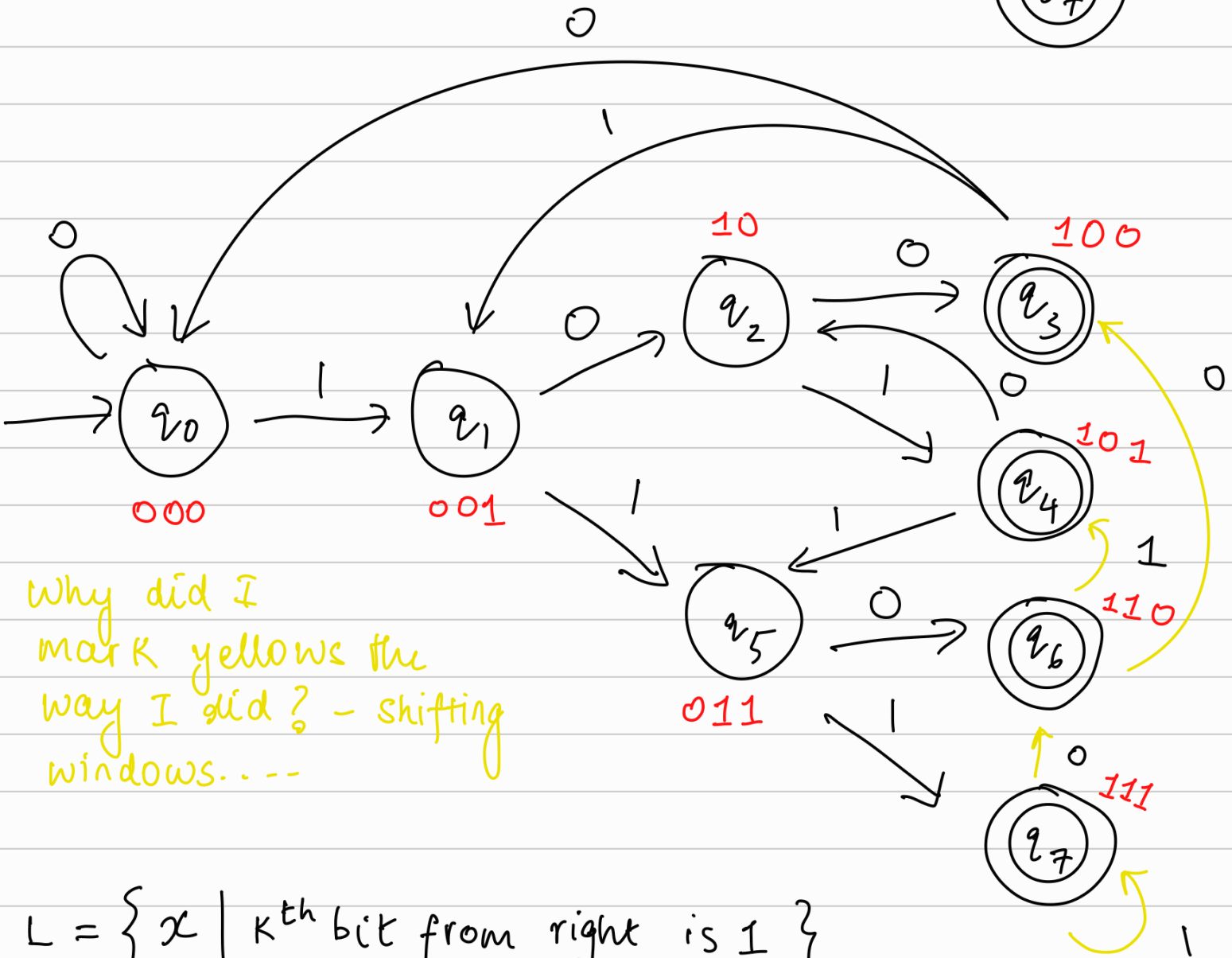
is this correct? No, as 1110 is rejected. the intuition is: q_0 is the 3rd last fellow....

Can we get an 'n-K' state NFA for an 'n' state DFA??? explore....

P.T.O.



Sir's initial build.



Why did I mark yellows the way I did? - shifting windows....

$$L = \{ x \mid K^{th} \text{ bit from right is } 1 \}$$

↳ requires 2^K states.

REGULAR EXPRESSIONS (RE) {rep. ∞ string in finite form}

* Algebraic representation.
* expresses computing power of FA.

(e.g.) $L(M) = \{0\} \rightarrow RE = 0$.

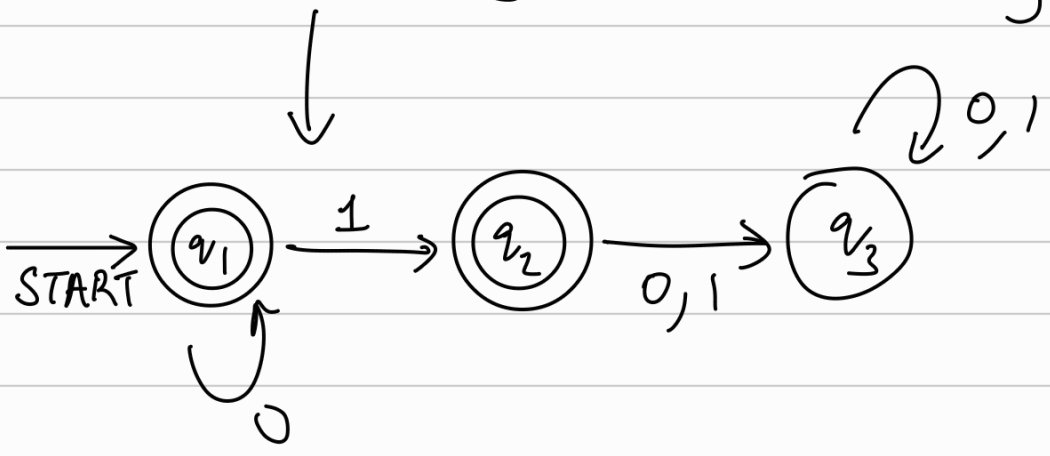
$L(M) = \{\epsilon, 0, 00, 000, \ldots\} \rightarrow RE = 0^*$

0^* ↓ zero or more 0's.

RE_1 0 RE_2 1 | let $RE_3 = 0 + 1$.

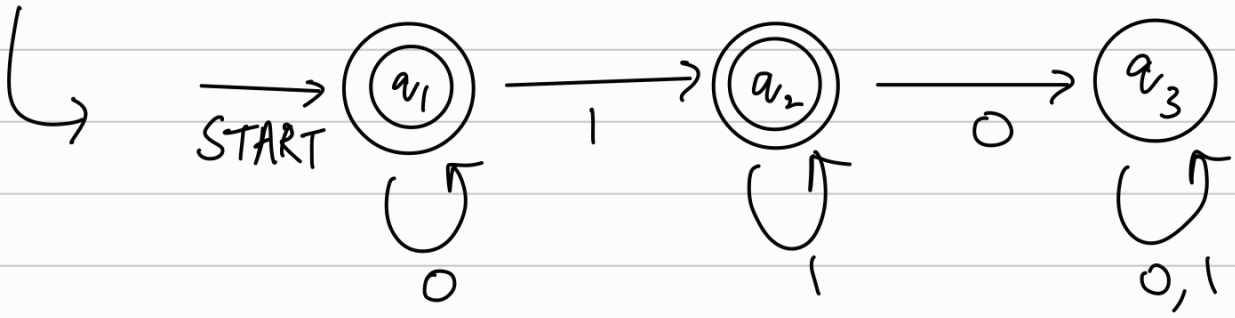
$\therefore L(RE_3) = \{0, 1\}$

$RE_4 = 0^* + 1 \rightarrow L(RE_3) = \{\epsilon, 0, 00, \ldots, 1\}$



$0^* + 1^*$

THIS IS WRONG. WHY?



$(0+1)^*$ → zero or more copies of $(0+1)$

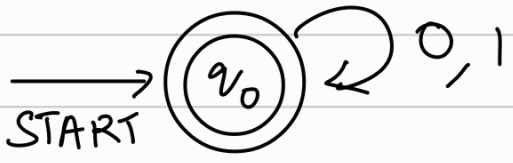
$\{\in, (0+1), (0+1)(0+1), (0+1)(0+1)(0+1), \ldots\}$

concat. on prev. result.

Obs: $L = \{0,1\}^*$.

Basically all possible bin strings of varying length!

1 → 0, 1
2 → 00, 01, 10, 11
& so on ...



$(0^*+1^*)^*$ → my thought: isn't this same as prev?

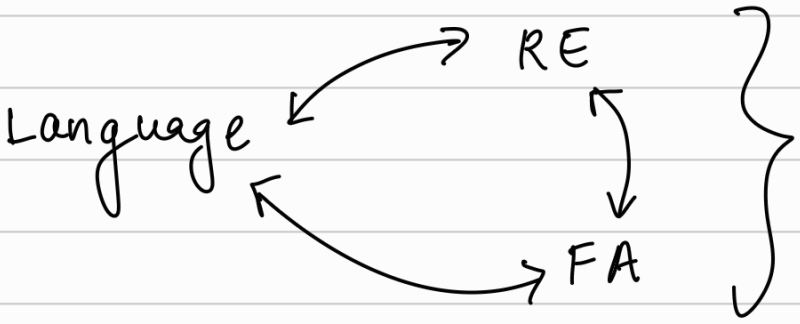
$\{\in, 0^*+1^*, (0^*+1^*)(0^*+1^*), (0^*+1^*)(0^*+1^*)(0^*+1^*), \ldots\}$

∴, $(0+1)^* \equiv (0^*+1^*)^*$

$L_1 = \{x \mid x$ contains 101 as substring$\}$

$L_1 = \underbrace{(0+1)^*}_{RE_1} \underbrace{1\,0\,1}_{RE_{2,3,4}} \underbrace{(0+1)^*}_{RE_5}$

$RE = RE_1 \; RE_{2,3,4} \; RE_5$ //



GOAL

$L_2 = \{ x \mid 3^{rd}$ symbol from right is 1 $\}$

$RE(L_2) = (0+1)^* 1 (0+1)(0+1)$

$RE_1 \quad RE_2 \quad RE_3 \quad RE_4$

3rd from right

$L_3 = \{ x \mid \#_{1's}(x)$ is divisible by 5 $\}$

$0^*10^*10^*10^*10^*1$ } ?

$(0^*10^*10^*10^*10^*10^*)^* + 0^*$

↳ 0.... 0 is a valid string.

↓

this pattern takes care of multiple of 5 1's.

$L = \{ x \in \{0,1\}^* \mid$ decimal_equivalent $(x) \% 3 = 0 \}$

try getting DFA, NFA, NFA with ϵ, RE.

given $L \subseteq \Sigma^*$, can we find (i) RE for L
(ii) DFA for L
(iii) NFA for L.

NFA with Epsilon

DFA → $\Sigma = \{0,1\}$, $\delta: Q \times \Sigma \to Q$
NFA → $\Sigma = \{0,1\}$, $\delta: Q \times \Sigma \to 2^Q$

↓

can NFA move read control without reading i/p symbol

↳ does this increase computational power of FA.

$010 \equiv \epsilon 0 \epsilon \epsilon \cdots \epsilon 10$

for a symbol a, $\epsilon^* a = a\epsilon^* = \epsilon^* a \epsilon^* = a$

let $\Sigma = \{0, 1, \epsilon\}$ or $\{0, 1, \lambda\}$ $(\lambda = \epsilon)$

↓

empty symbol.

although $\epsilon^* a \epsilon^*$ is equal to a, they aren't computationally equal, as in $\epsilon^* a \epsilon^*$ many states might be traversed.

if ϵ → empty string, then use $\hat{\delta}(q_0, \epsilon)$
→ empty symbol, then use $\hat{\delta}(q_0, \epsilon)$

does ∃ DFA for each ϵ-NFA ?

(basically plonking ϵ between any 2 consecutive symbols to possibly acheive something greater ??)

↳ what is the advantage ? is not reading anything helpful?

for $a \in \Sigma$,

$\delta^D(q, a)$ → deterministic / no uncertainty

$\delta^N(q, a)$ → uncertainty ↘ $\epsilon^* a \epsilon^*$

↓

be anything

$\epsilon a \epsilon \epsilon$

$\epsilon \epsilon a \epsilon$

a

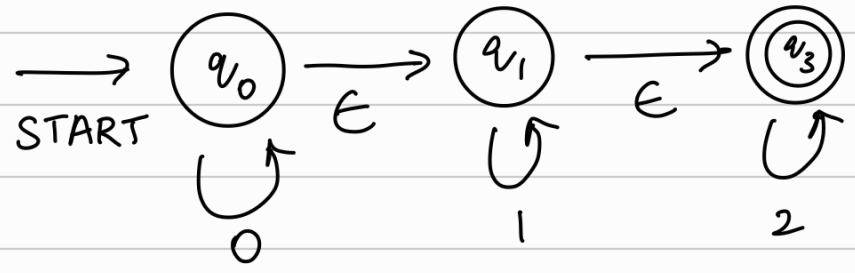
$\epsilon \epsilon \epsilon a \ldots$

& so on...

$L = (0+1)^*$ · does ∃ 2 DFA's ?

if not, p.t. ∄! DFA.

M:



↓

$\epsilon \epsilon \epsilon = \epsilon$

$0 \epsilon \epsilon = 0$

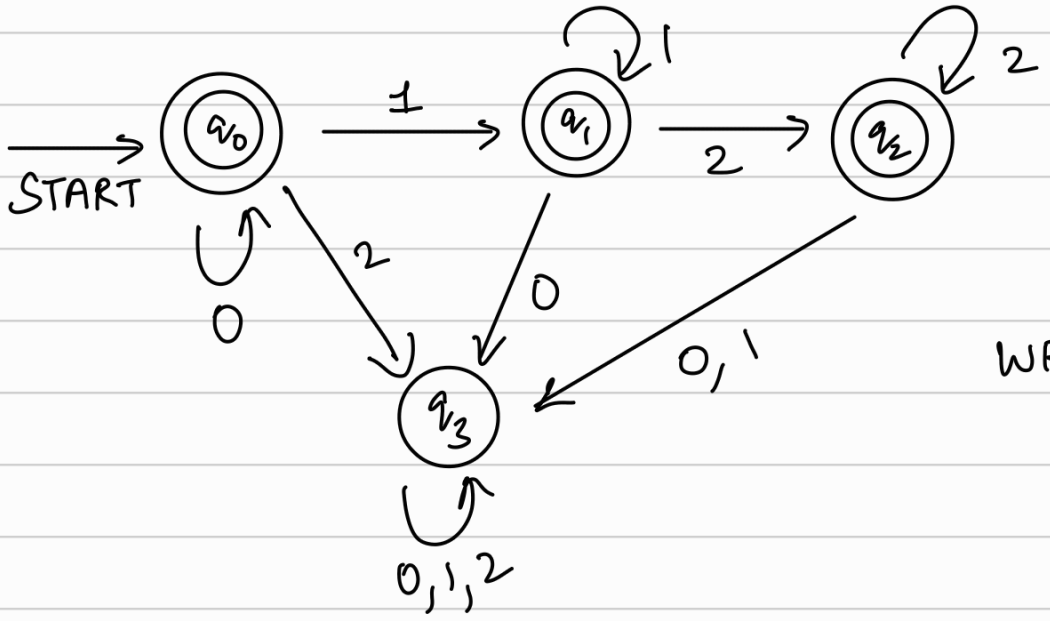
$\epsilon 1 \epsilon = 1$

$\epsilon \epsilon 2 = 2$

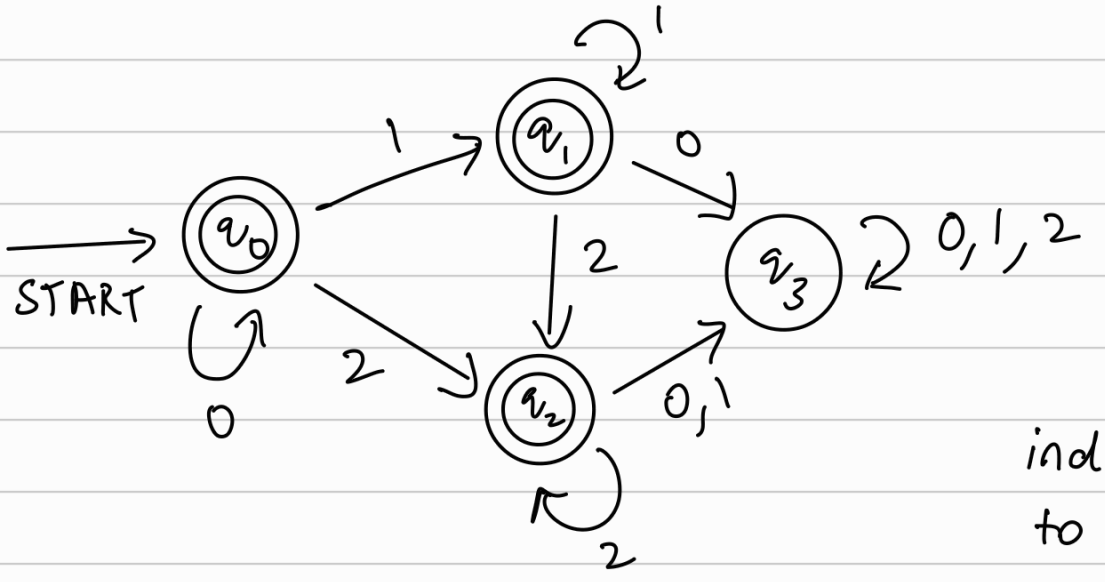
$00 \epsilon 1 \epsilon 2 = 0012$

$00 \epsilon 222 = 00222$.

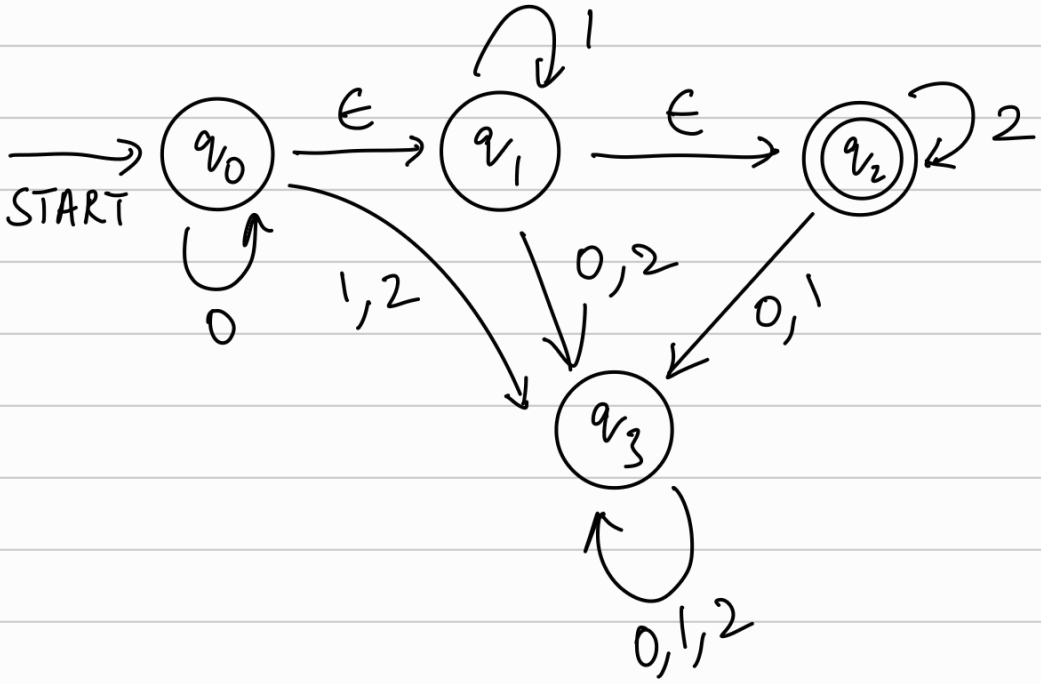
my obs: ϵ lets you pick what you want (or) lets you ignore what you want....



first iteration.
WRONG.
2 is not accepted.



2nd iteration.
indeed equivalent to ϵ-NFA - worked on 2 accepting ....



Epsilon Closure:

ϵ-closure of q is the set of all q' s.t. $q' \in Q$ and q' is reachable from q by reading one/more ϵ's.

$\forall\, q \in Q$, q ϵ-closure of q, $E(q)$

$E(q_1) = \{q_1, q_2\}$ $\quad$ $E(q_0) = \{q_0, q_1, q_2\}$

$E(q_2) = \{q_2\}$

$\delta : Q \times \Sigma \rightarrow Q$ | $\delta(q_0, 0) \leftrightarrow \delta(q_0, 0)$, $\delta(q_0, \epsilon^* 0)$, $\delta(q_0, \epsilon^* 0 \epsilon^*)$

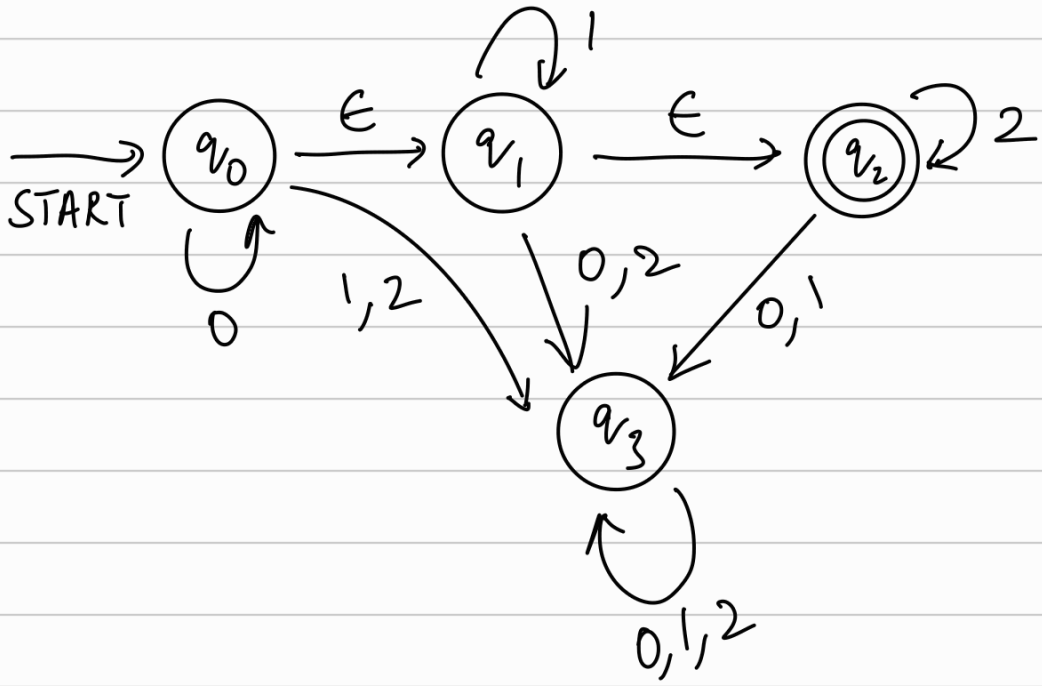
for ϵ-NFA.

$L = \{ x \in \{0,1\}^* \mid \#_0(x)$: ODD, $\#_1(x)$: EVEN $\}$ → think of RE for this .....

"CAN WE construct an NFA from ϵ-NFA?"

↳ in other words, how to bring in non-determinism using $a \in \Sigma$ replacing ϵ.

recall ϵ-closure: reach states using sequence of ϵ's. Our goal is to acheive the same, w.o. ϵs. for the given e.g.;



how do I replace the ϵ-s with extra state transitions? give a pathway to reach the ϵ-closure states & more using the a's $\in \Sigma$.

∃ an ϵ-path b/w q_0 and q_1, and q_1 and q_2. this enables the machine to reach both q_1 & q_2 w.o. reading anything. What does this actually mean? can we replace the ϵ with others?

$$\delta(q_0, 0) \equiv \hat{\delta}(q_0, 0\epsilon^*), \hat{\delta}(q_0, \epsilon^* 0), \hat{\delta}(q_0, \epsilon^* 0 \epsilon^*)$$

Consider $0\epsilon \rightarrow$ you reach q_1 $\quad 0 \rightarrow$ stay at q_0

$\epsilon 0 \rightarrow$ you reach q_3

$0\epsilon\epsilon \rightarrow$ you reach q_2

$\therefore \delta(q_0, 0) = \{q_0, q_1, q_2, q_3\}$

$$\delta(q_0, 1) \equiv \hat{\delta}(q_0, \epsilon^*1), \hat{\delta}(q_0, 1\epsilon^*), \hat{\delta}(q_0, \epsilon^*1\epsilon^*)$$

$q_0, 1 \rightarrow q_3$
$q_0, \epsilon 1 \rightarrow q_1$
$q_0, \epsilon\epsilon 1 \rightarrow q_3$
$q_0, \epsilon 1\epsilon \rightarrow q_2$

$\therefore \delta(q_0, 1) = \{q_1, q_2, q_3\}$

$$\delta(q_0, 2) \equiv \hat{\delta}(q_0, \epsilon^*2), \hat{\delta}(q_0, 2\epsilon^*), \hat{\delta}(q_0, \epsilon^*2\epsilon^*)$$

$q_0, 2 \rightarrow q_3$
$q_0, \epsilon\epsilon 2 \rightarrow q_2$

$\rightarrow \delta(q_0, 2) = \{q_2, q_3\}$

So, $\delta(q_0, 0) = \{q_0, q_1, q_2, q_3\}$

$\delta(q_0, 1) = \{q_1, q_2, q_3\}$

$\delta(q_0, 2) = \{q_2, q_3\}$

Similarly,

$$\delta(q_1, 0) \equiv \hat{\delta}(q_0, \epsilon^*0\epsilon^*)$$

$q_1, 0 \rightarrow q_3$
$q_1, \epsilon 0 \rightarrow q_3$

$\therefore \delta(q_1, 0) = q_3$

$$\delta(q_1, 1) \equiv \hat{\delta}(q_1, \epsilon^* 1 \epsilon^*)$$

$q_1, 1 \to q_1$
$q_1, \epsilon 1 \to q_3$
$q_1, 1\epsilon \to q_2$
$\}$ $\to$ $\therefore$, $\delta(q_1, 1) = \{q_1, q_2, q_3\}$

$$\delta(q_1, 2) \equiv \hat{\delta}(q_1, \epsilon^* 2 \epsilon^*)$$

$q_1, 2 \to q_3$
$q_1, \epsilon 2 \to q_2$
$\}$ $\to$ $\therefore$, $\delta(q_1, 2) = \{q_2, q_3\}$

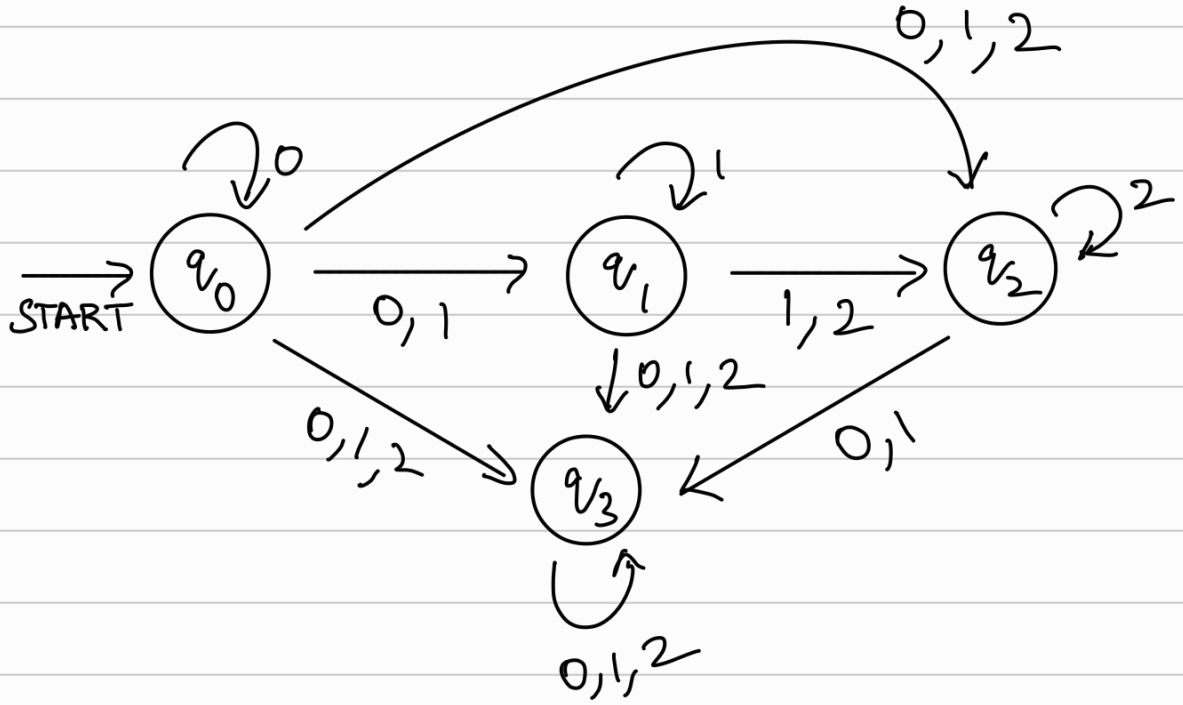
sim $\to$ $\delta(q_2, 0) = \{q_3\}$

$\delta(q_2, 1) = \{q_3\}$

$\delta(q_2, 2) = \{q_2\}$

$q_3 \to$ anyways $\forall a \in \Sigma$ belongs to q_3 itself.

$\therefore$, updated automaton using prev info:



next qs: what about final states?
we were able to reach q_2 in ϵ-NFA from q_0 & q_1 → ∴, q_0, q_1 & q_2 must be the final states here.....
↳ accounting for lack of ϵ by defining this.

formalizing this:

let M_1 be my ϵ-less NFA / NFA w.o. ϵ.

$$M_1 = \{Q_1 = \{q_0, q_1, q_2, q_3\}, \delta_1, \Sigma, q_0, F_1\}$$

$$\Downarrow$$

$$M = \{Q = \{q_0, q_1, q_2, q_3\}, \delta, \Sigma, q_0, F\}$$

↳ ϵ-NFA.

then, for my M_1:

$\delta_1(q_0, a)$ → represents what all states I could've possibly REACHED in the ϵ-NFA reading a at q_0, using ϵ's. if I can reach elements in ϵ-closure with ϵ's, then I can reach the state after they read a & it's ϵ-closure too!

as $\epsilon^* a \epsilon^* = a$ ←

∴, $\delta_1(q_0, 0) = E(\delta(E(q_0), 0))$

checks: $\delta_1(q_0, 0) = E\left(\delta\left(E(q_0), 0\right)\right)$

$= E\left(\delta\left(\{q_0, q_1, q_2, q_3\}, 0\right)\right)$

$= E\left(\delta(q_0, 0) \vee \delta(q_1, 0) \vee \delta(q_2, 0) \vee \delta(q_3, 0)\right)$

$= E\left(\{q_0, q_3\}\right)$

$= \{q_0, q_1, q_2, q_3\}$ → agrees with prev results

$\delta_1(q_0, 1) = E\left(\delta\left(E(q_0), 1\right)\right)$

$= E\left(\delta\left(\{q_0, q_1, q_2, q_3\}, 1\right)\right)$

$= E\left(\delta(q_0, 1) \vee \delta(q_1, 1) \vee \delta(q_2, 1) \vee \delta(q_3, 1)\right)$

$= E\left(\{q_3, q_1\}\right)$

$= \{q_1, q_2, q_3\}$ → again, agrees //

∴, for M_1 (ϵ-less equivalent of M),

$$\delta_1(q, a) = E\left(\delta\left(E(q), a\right)\right).$$

and final states in ϵ-less = states whose ϵ-closure contains final state in ϵ-NFA {because u could reach final state from them}

$\therefore$, $F_1 = \{ q \in Q \mid q_f \in E(q), q_f \in F \}$

if $E(q) \cap F \neq \phi$, then q is final state ϵ-less NFA.

"if min DFA has n states, can we get a $\leq n$ state NFA / ϵ-NFA, w.o. losing computational power?"

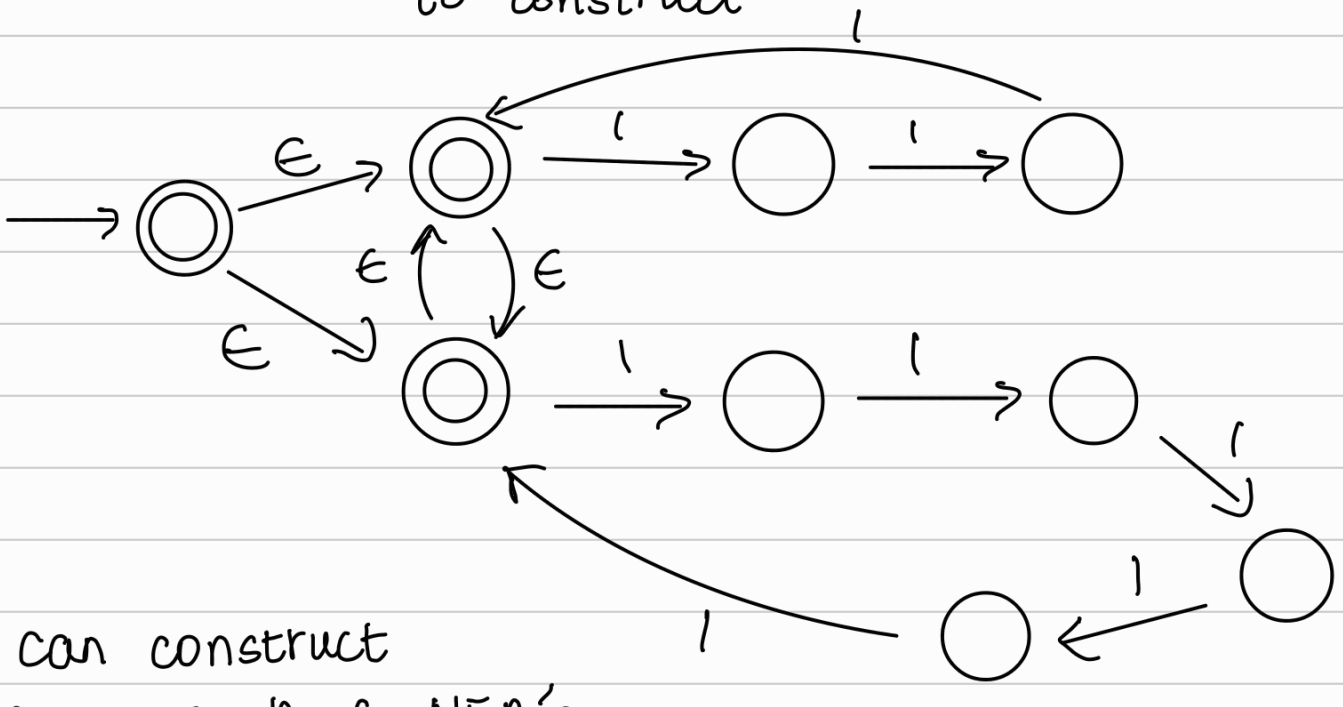
recall Assignment-1, Q7 → we got a 5 state NFA accepting RE $(11111 + 111)^*$. can we get <5 state ϵ-NFA / NFA?

min DFA has 9 states for this RE {check solns.}
↳ also proof for this? shall be discovered.

design power: ϵ-NFA > NFA > DFA

↓ easiest to construct

↓ hardest.

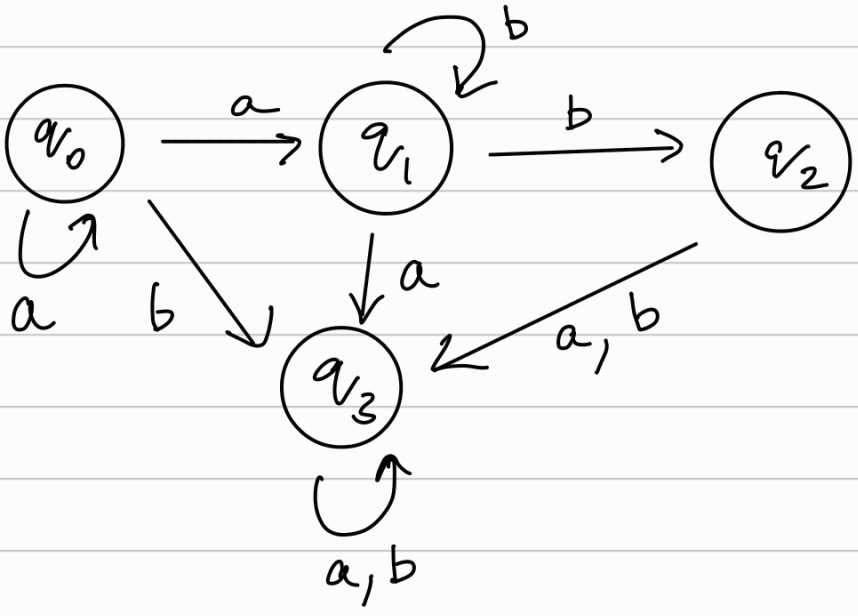


can construct many such ϵ-NFA's

given an RE, construct ϵ-NFA first, then NFA s.t. $L(NFA) = L(RE)$. if we construct an RE from NFA, then $RE \equiv NFA \equiv \epsilon\text{-NFA}$.

Now, how do we convert an NFA into a DFA?

consider:



in a DFA, there's no uncertainty wrt next state. Here,

$$\delta(q_0, a) = \{q_1, q_3\} \quad | \quad \delta(q_1, a) = \{q_3\}$$

$$\delta(q_0, b) = \{q_3\} \quad | \quad \delta(q_1, b) = \{q_1, q_2\}$$

$$\delta(q_2, a) \;\&\; \delta(q_2, b) = \{q_3\}.$$

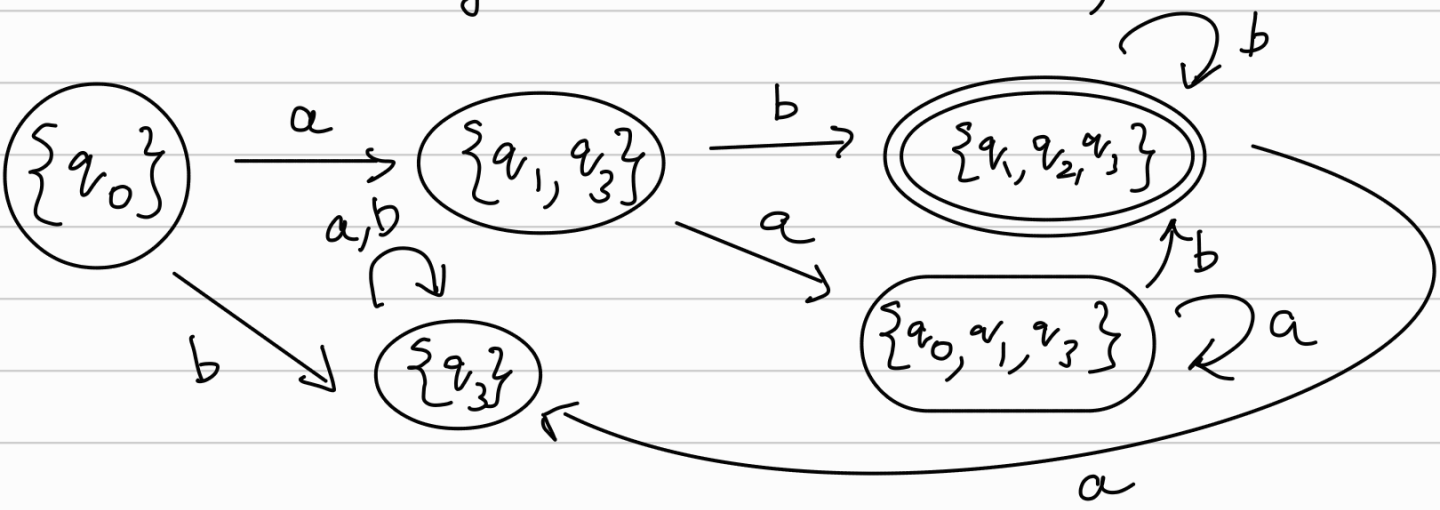
We read alphabets (a,b) for the obtained sets & observe what "set" of states those reach.

$$\delta_D(\{q_1, q_3\}, a) = \delta(q_1, a) \vee \delta(q_3, a) = \{q_3\}$$

& so on & then, consider the obtained sets as the states themselves → eradicates non-determinism.

What is the final state? ⟶ any state containing the final state of NFA, is a final state. Hence,

wkt. eventually q_2 will be reached if iterating at this state



Q: is the constructed automaton minimum?