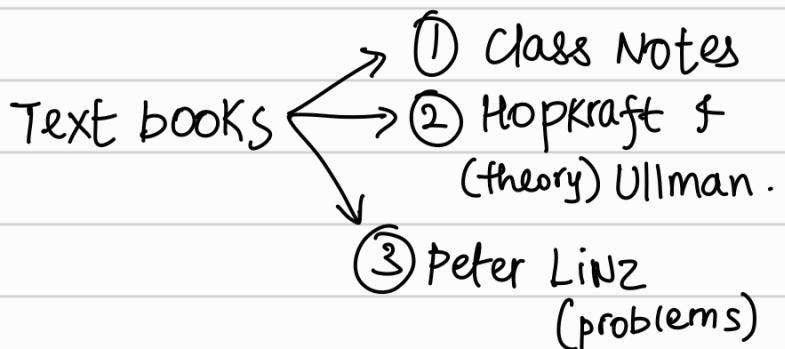


first words spoken → write algorithm / recipe to prepare coconut chutney based on DAA course done earlier (xD)

split:

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



Purpose of studying TOC

(actually why study at all?
go to ISKCON 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 I K 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 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, PLD, 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 achieving purpose
- humans → natural language
 - machines → digital language (voltages) $\xrightarrow{+5V}$ $\xrightarrow{0V}$

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

($q_1 \rightarrow q_2 \rightarrow \dots \rightarrow q_{opt}$, $q_{opt} \rightarrow$ 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.

there's an infinite tape divided into cells - each cell $\rightarrow \{0,1\}$



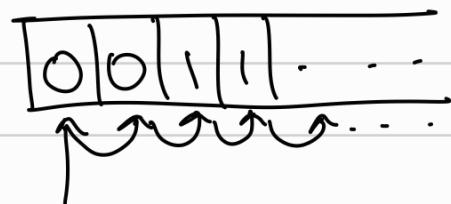
the machine can:

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

the machine can't:

- ↳ jump cells
- ↳ read words
- ↳ write
- ↳ store words

α-bet : $\Sigma = \{0, 1\}$



READ & MOVE

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

? : does M accept x

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

can we think of FSA to recognise 'S'

i/p: x $x \in S - \text{Yes}$
 $x \notin S$

wrt $S \rightarrow$ strings ending with 01

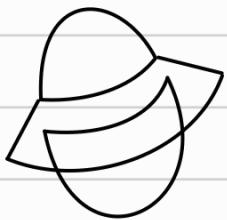
i/p: $S \subseteq \Sigma^* +$ any string in S ending with 01

design FSA \rightarrow to recognise/accept S .

$$S = \text{prefix} + 01$$

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

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

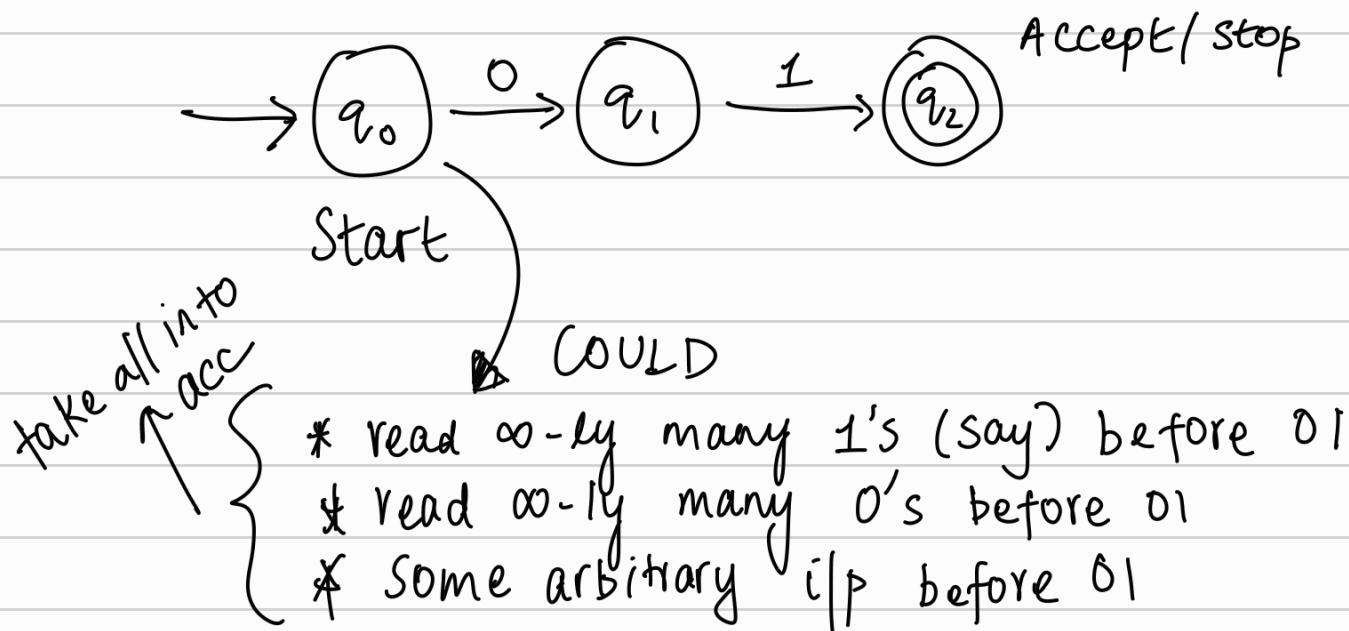


LOWER BOUND THEORY : min T.C. reqd. to
solve. (like sorting - $n \log n$)
 $(\Omega(f(n)))$

α -bet \rightarrow set of symbols.

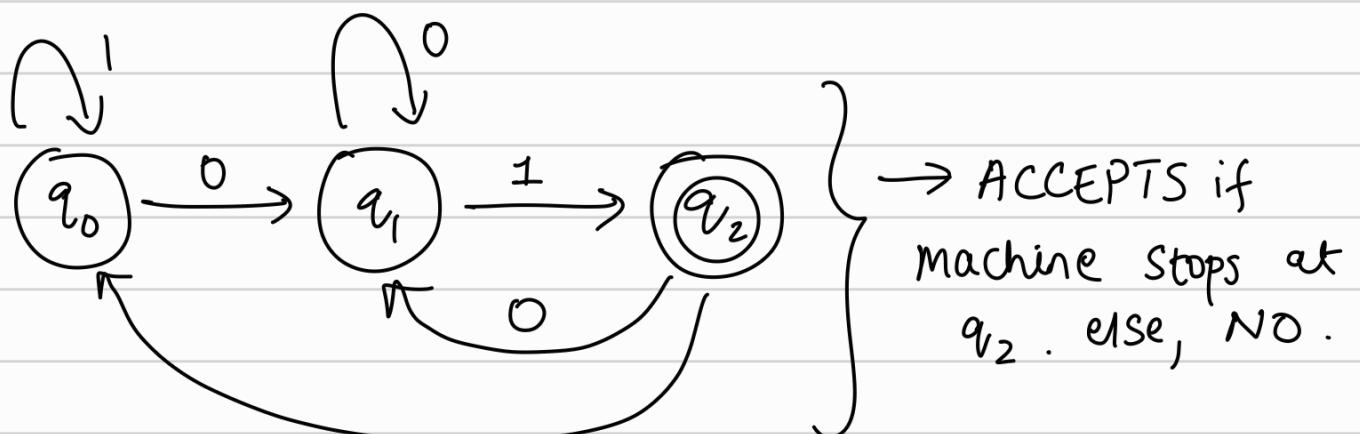
somehow store the prefix in the FSA.

States - talk about intermediate computations



* 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.



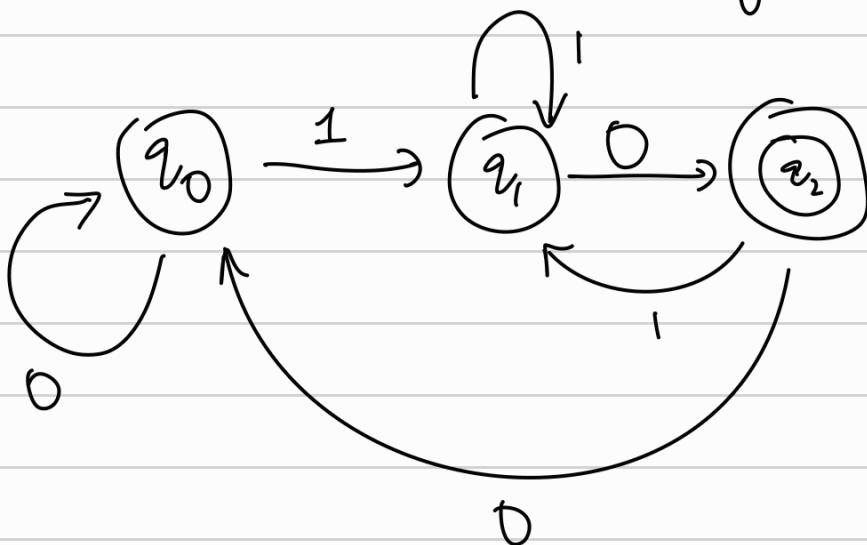
TRY for all i/p's. ↓ FSA.

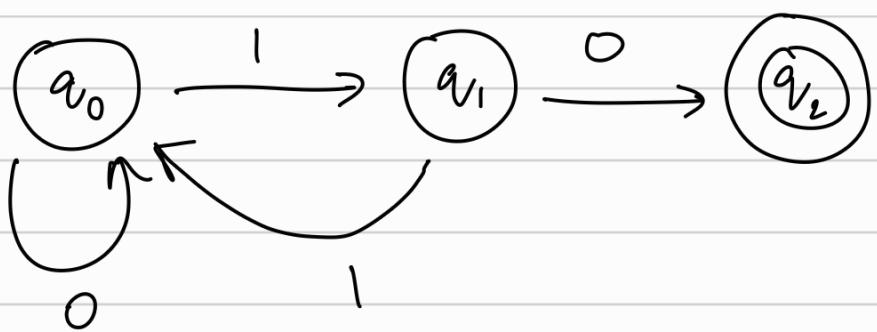
Simulate Σ on FSA. if FSA stops at q_2 , ACCEPT.
else, REJECT

Stopping at
 q_0 or q_1

- Set of all strings ending with 10
- Set of all strings containing 10

a)





The diagram shows the formal definition of an FSA as a tuple $(Q, \Sigma, \delta, q_0, F)$. Annotations with arrows point to each component: 'set of states' points to Q , 'i/p.' points to Σ , 'transition fn.' points to δ , 'start state' points to q_0 , and 'final state' points to F .

FSA \rightarrow 5-tuple machine

$\delta : \text{DOMAIN} \rightarrow \text{CO-DOMAIN.}$

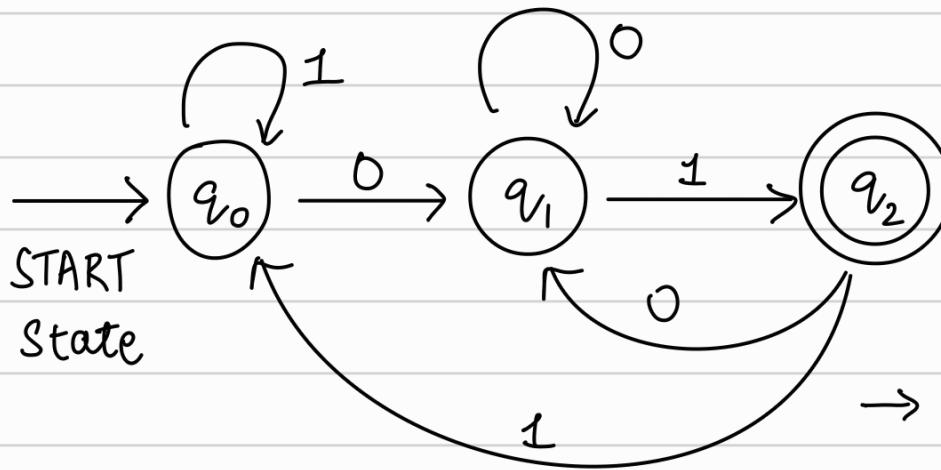
$Q \times \Sigma \xrightarrow{\quad} Q$

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

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

$\underbrace{0, 0}_{\downarrow}, \underbrace{0, 1}_{\downarrow}, \underbrace{0, 1}_{\downarrow}, \underbrace{1, 0}_{\downarrow}, \underbrace{1, 1}_{\downarrow}$, and so on

$x \rightarrow \dots 01.$



$$Q = \{q_0, q_1, q_2\}$$

$$\Sigma = \{0, 1\}$$

$$\downarrow$$

$$\{q_0, q_1, q_2\}$$

→ FINITE STATE DIAG

$$\delta: (Q \times \Sigma) \rightarrow Q$$

$$\text{e.g. } (q_0, 1) \rightarrow q_0$$

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

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 :

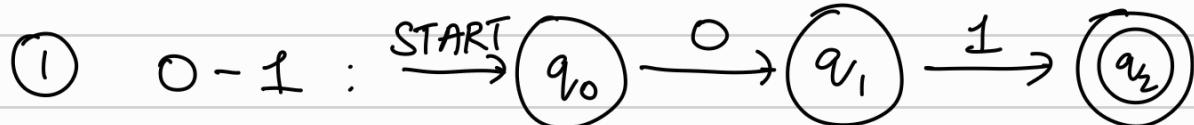
↳ just accept 01

↳ accept len=3 + ends with 01

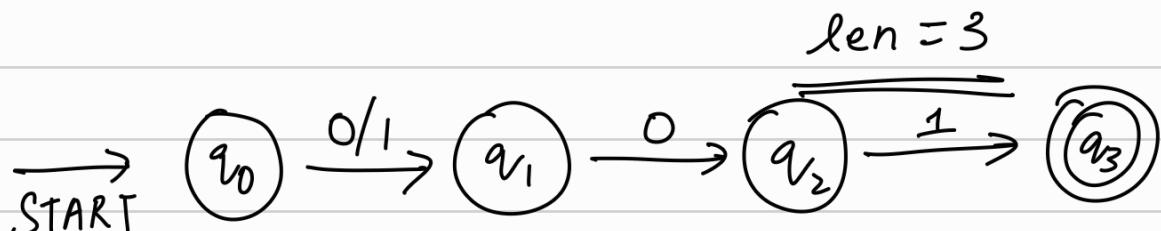
↳ n len=4 + ends with 01

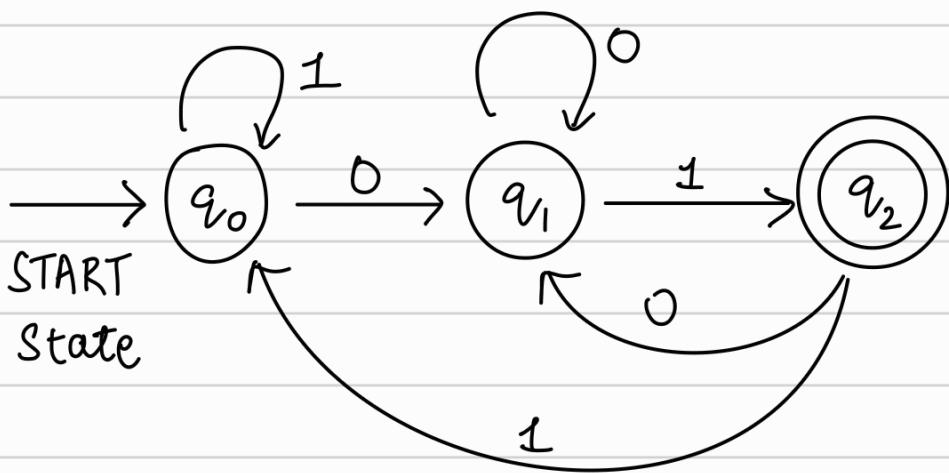
; ; ; ;

↳ reject strings not ending with 01.



② 0-1 + len = 3





$$Q = \{q_0, q_1, q_2\}$$

$$\Sigma = \{0, 1\}$$

$$\downarrow$$

$$\{q_0, q_1, q_2\}$$

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

MUST ACCEPT ALL VALID STRINGS (end with 01)

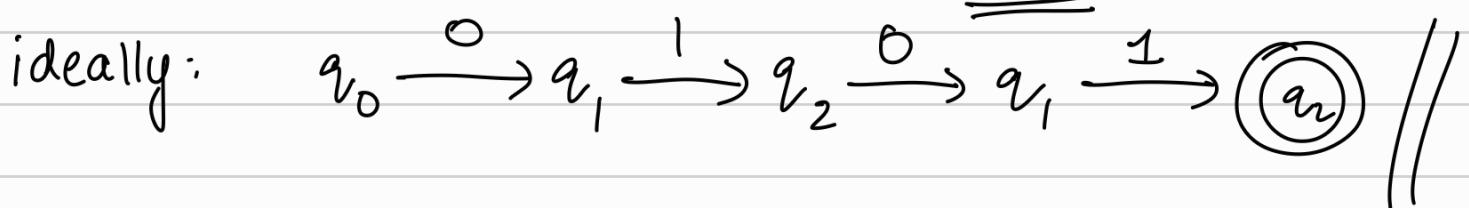
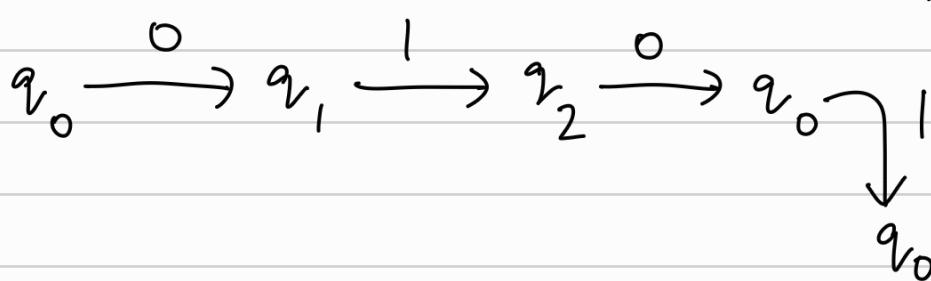
FSA

MUST REJECT ALL INVALID STRINGS (not ending with 01)

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

instead of q_1

if $(q_2, 0)$ goes to q_0 , 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



"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!
 $\delta(q_2, 1) = q_0 \rightarrow$ then things work . . .

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

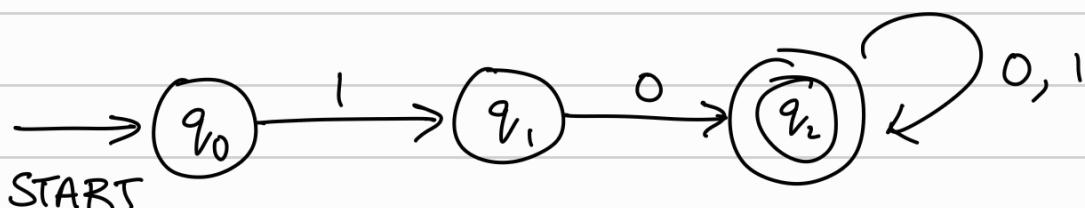
$$\text{FSA} = \left\{ Q = \{q_0, q_1, q_2\}, \Sigma = \{0, 1\}, \delta, q_0, F = \{q_2\} \right\}$$

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

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

$$\frac{10}{l=2}, \frac{100, 101}{l=3}, \frac{1001, 1010, 1011}{l=4}, 1000, \text{ and so on . . .}$$

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



defined: $\delta(q_0, 1), \delta(q_1, 0), \delta(q_2, 1)$

and $\delta(q_2, 0)$. Others?

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

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

is δ_M well defined?

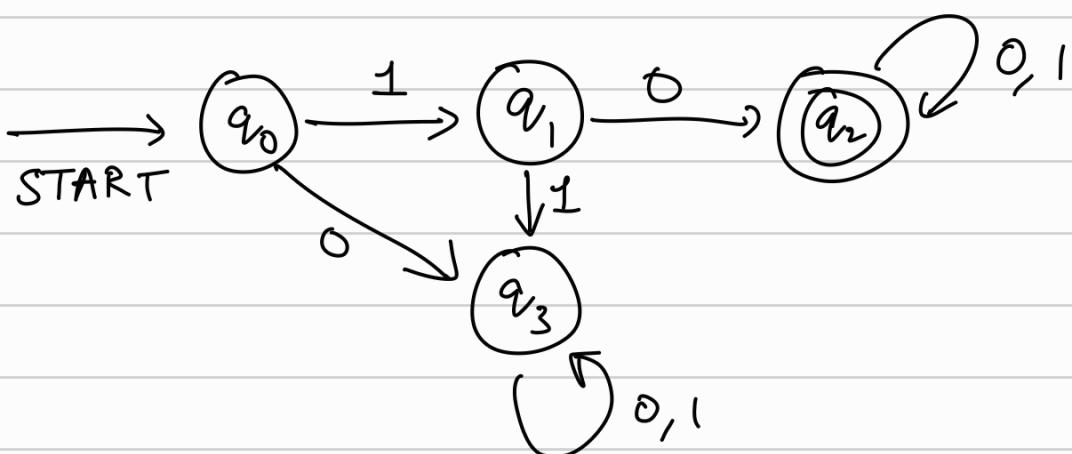
$$\begin{aligned} (q_0, 0) &\rightarrow q_3 & \text{INVALID} \\ (q_0, 1) &\rightarrow q_1 \\ (q_1, 0) &\rightarrow q_2 \\ (q_1, 1) &\rightarrow q_3 & \text{INVALID} \\ (q_2, 0) &\rightarrow q_2 \\ (q_2, 1) &\rightarrow q_2 \end{aligned}$$

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

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

∴ updated M f diag:

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



$$\begin{cases} \delta(q_3, 0) = q_3 \\ \delta(q_3, 1) = q_3 \end{cases}$$

existence at q_3 itself
means disagreement.
i.e., exist there + symbols
after disagreement.

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

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

$\left. \begin{array}{l} \text{ending at ANY state} \\ \text{other than } q_2 \text{ is} \\ \text{REJECTION} \dots \end{array} \right\}$

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

1-state FSA :

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

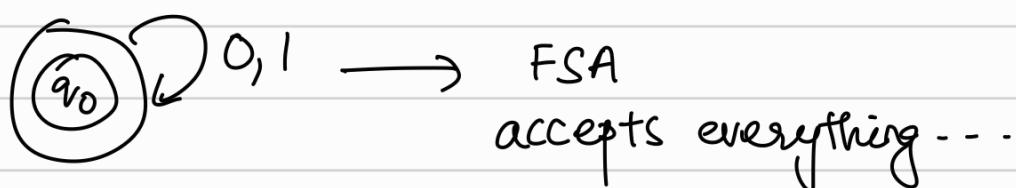
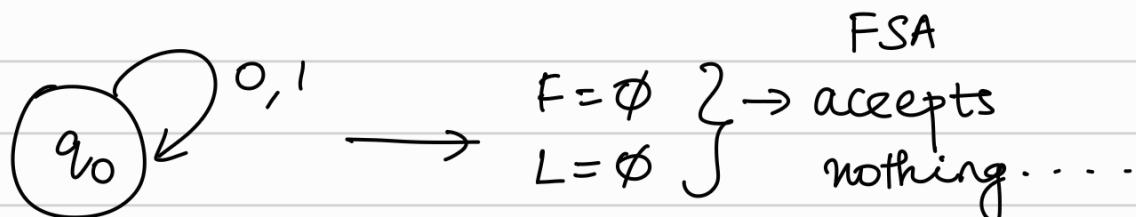
$$\left\{ \begin{array}{l} \downarrow \\ \{q_0\} \times \{0, 1\} \rightarrow \{q_0\} \end{array} \right.$$

$\left. \begin{array}{l} F \subseteq Q \text{ and not } F \in Q \\ \hookrightarrow \text{NEED NOT BE UNIQUE} \\ q_0 \in Q \rightarrow \text{UNIQUE} \end{array} \right\}$

$\delta \rightarrow$ 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 (?)
 $\hookrightarrow n(\text{languages}) = n(\text{subsets}) = 2^n$.
 $n \rightarrow \text{no. of final states}$
- * problem is solved \rightarrow if language identified for each state (?)



$$L \subseteq \{\epsilon, 0, 1, 00, \dots\}$$

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

$\left. \begin{array}{l} \text{didn't understand} \\ \text{language part well.} \\ \text{hope to understand} \\ \text{as course} \\ \text{progresses.} \end{array} \right\}$

$x - \xrightarrow{} x - \xrightarrow{} x$

30th Jan -

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

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

language \rightarrow set of all strings accepted by M.

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

M_1 accept L
 reject Σ^*/L

$q_0 \xrightarrow[x]{\quad} q_f$ accepted state.

$q_0 \xrightarrow[x]{\quad} q_f$ rejected state.

$L(M) \rightarrow$ 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 \exists a FSA " M " accepting S ?

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

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

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

↓
say ending with 01

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

$$\hat{\delta} : \hat{\delta}(q, x) \\ x \in \Sigma^* \quad x = x_1 x_2 x_3 \dots$$

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

$$= \hat{\delta}\left(\delta(q_0, x_1), x_2 x_3 \dots\right) = \hat{\delta}(q_1, x_2 x_3 \dots)$$

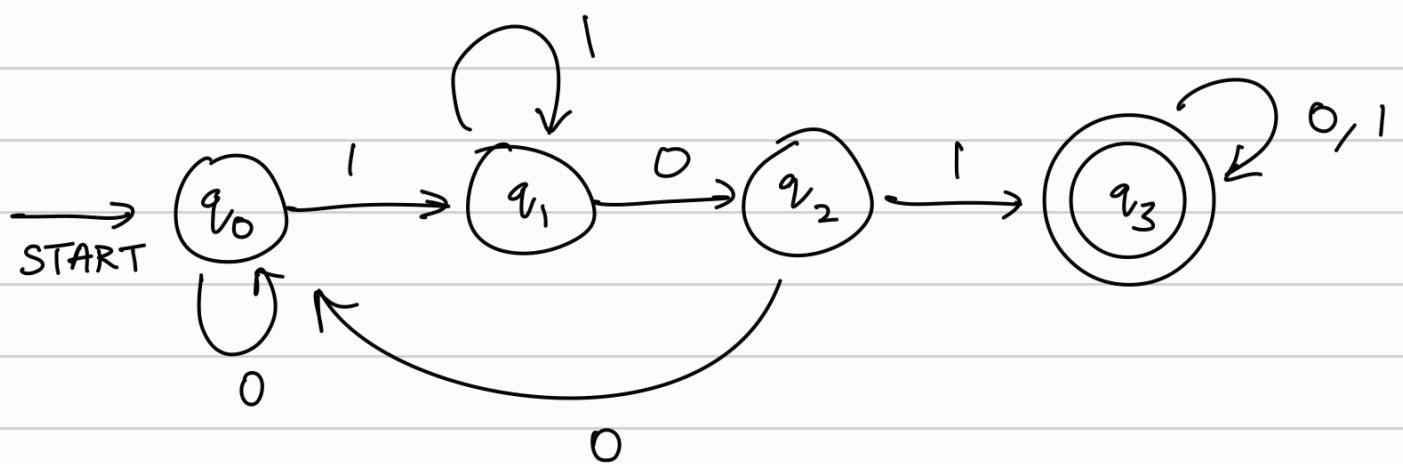
$$= \hat{\delta}\left(\delta(q_1, x_2), x_3 x_4 \dots\right) = \hat{\delta}(q_2, x_3 x_4 \dots)$$

and so on, until:

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

\$\hookrightarrow \\$ \quad \underline{\underline{\quad}}

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



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

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

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

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

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

$$\delta(q_1, 1) = q_1 \rightarrow \text{AND NOT } q_0, \text{ as } 1101 \uparrow$$

won't be accepted!

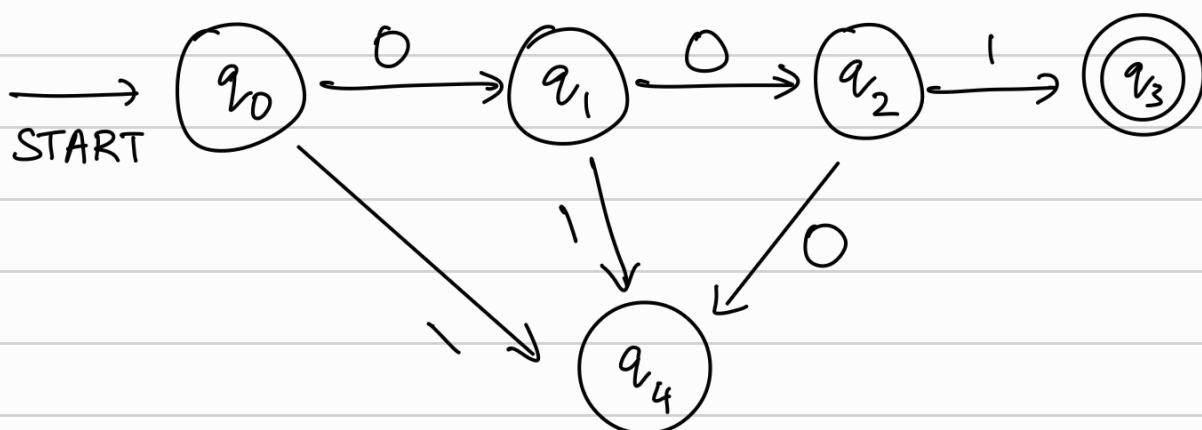
$\delta(q_2, 0) = q_0 \rightarrow$ AS the sequence breaks, so must search for 101 from first

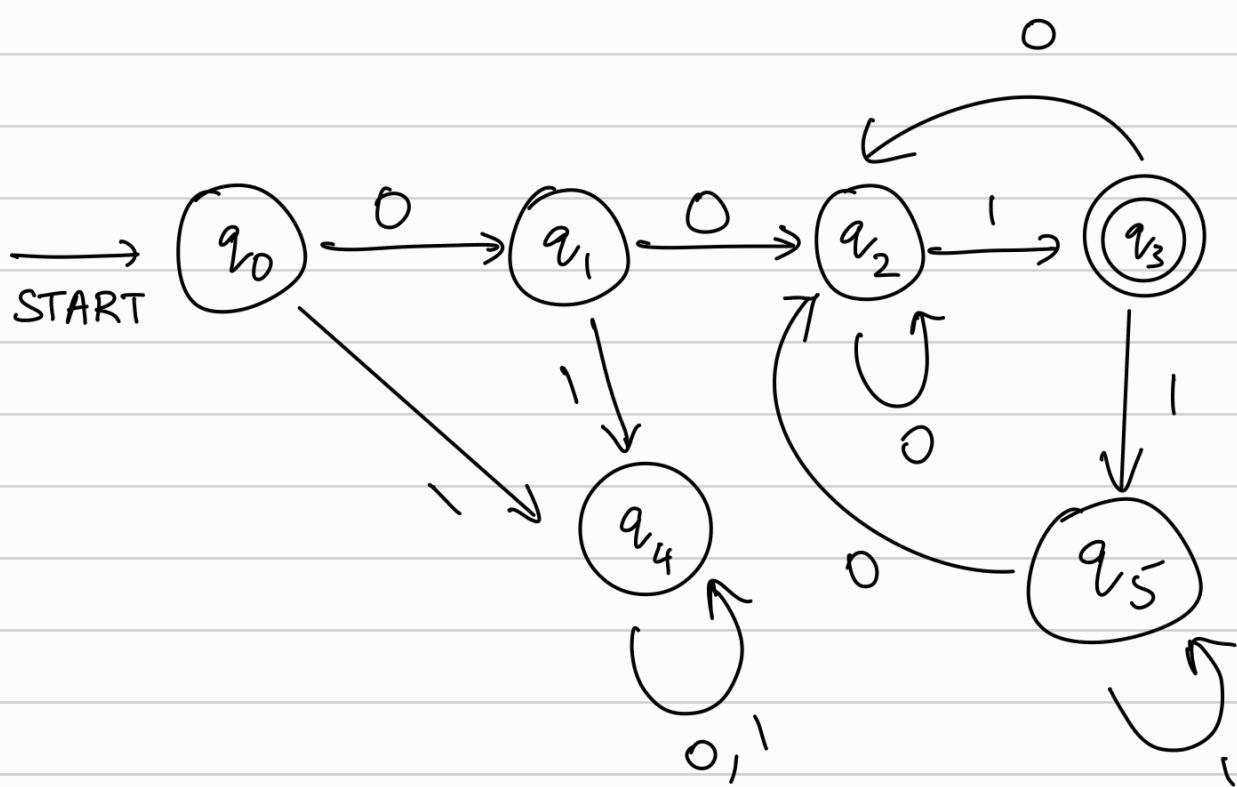
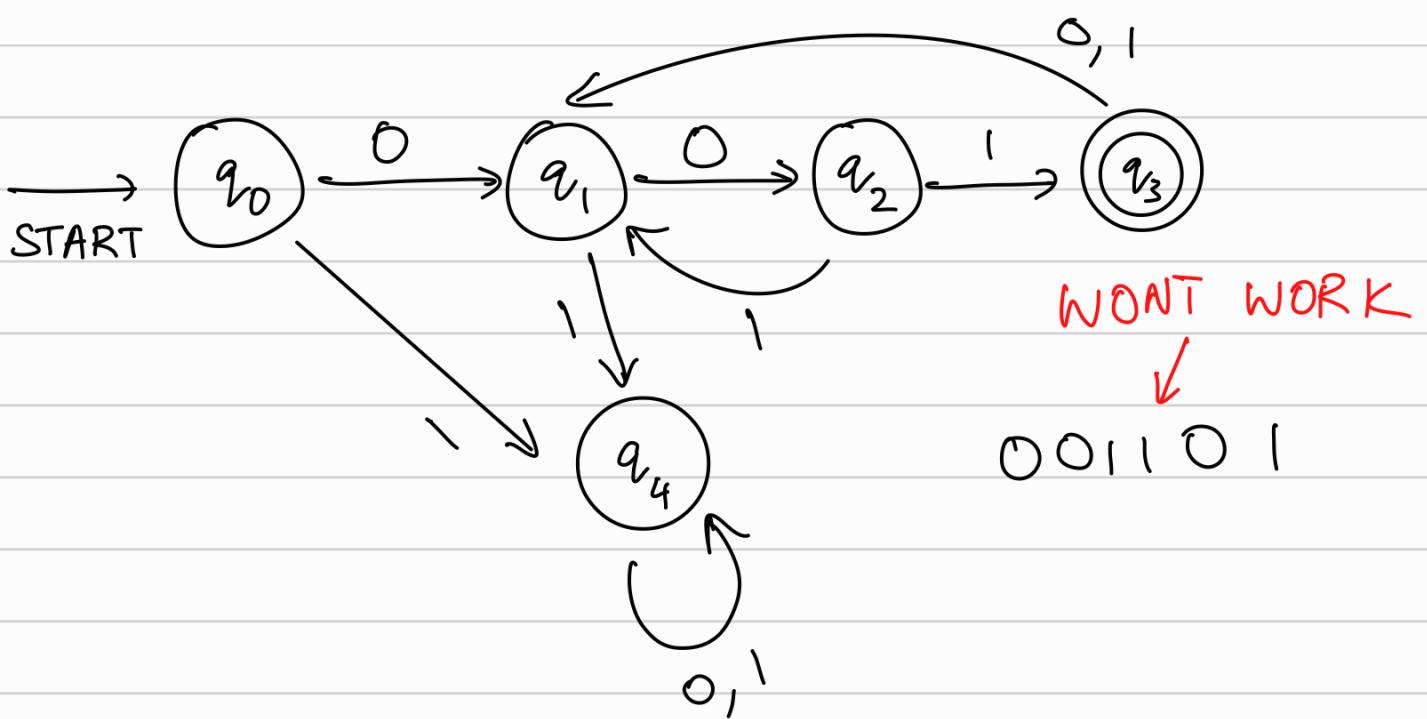
$$\delta(q_2, 1) = q_3 \rightarrow 101 \text{ discovered.}$$

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

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

try with 001 (then finetune - - -)





WHAT MISTAKES occurred :

↳ Self looping at q_4 at 001101 . . .
 ↳ shdn't go back to any of the prev states . . .

∴ introduce q_5 .

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

$$\delta(q_0, 1) = q_4 \text{ (REJECT)}$$

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

$$\delta(q_1, 1) = q_4 \text{ (REJECT)}$$

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

$$\delta(q_2, 0) = q_2 \text{ (Self-loop, in case many 0's occur - preserve sequence)}$$

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

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

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

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

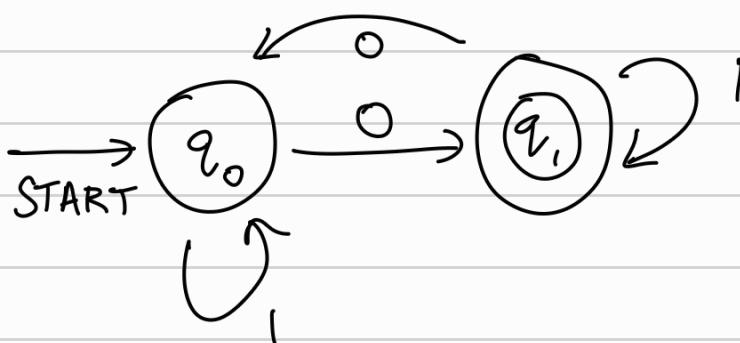
$$L_4 = \left\{ x \in \{0, 1\}^* \mid x \text{ has even 1's \& odd 0's} \right\}$$

Observations:

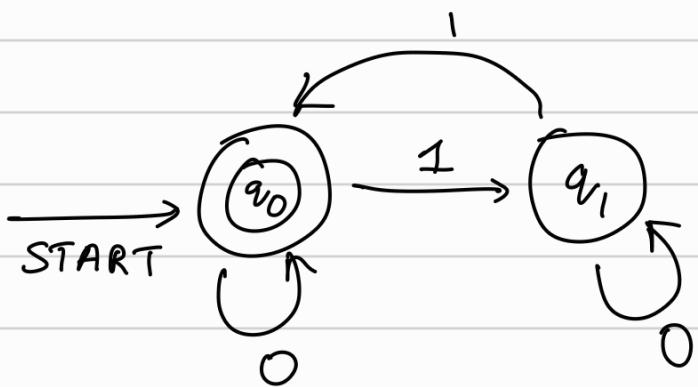
* x is ALWAYS of ODD length . . .

* self-loop DOESN'T aid in counting .

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



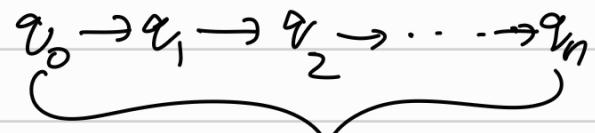
Q2) even no. of 1's \rightarrow no constraints on 0's.



Q3) How to unify both of these ???

$L_4 \rightarrow$ essentially odd #0's \wedge even #1's.

= $Q_1 \wedge Q_2 \dots$ questions



"Keep Asking Questions"

- SADDY :)

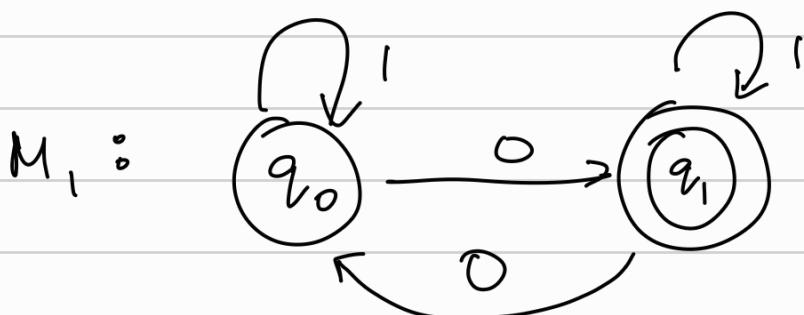
GREEDY

$q_n \rightarrow$ extraordinary

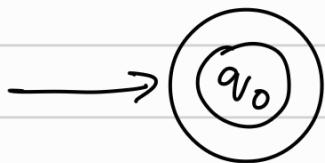
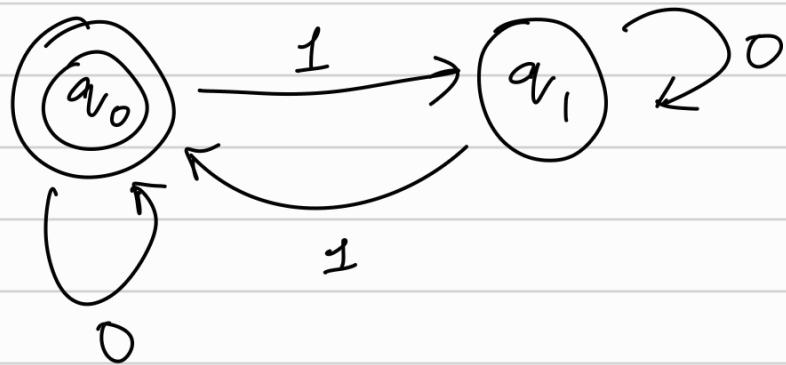
X ————— X ————— X

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

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



M_2 :

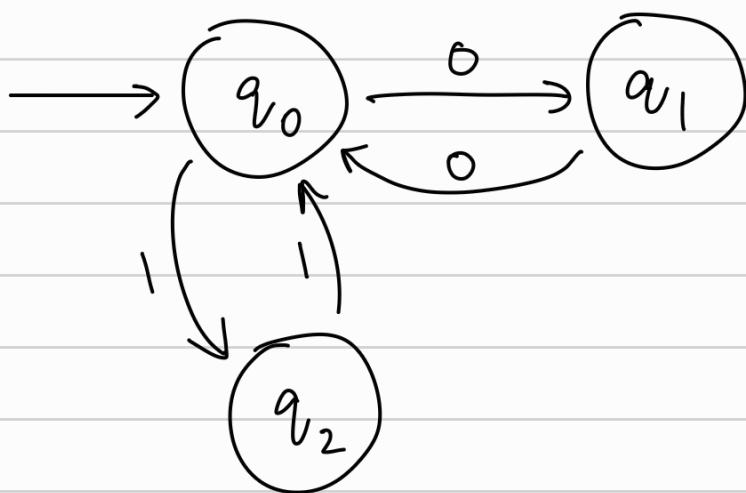


: FSA accepts a string without reading anything.

Accepts empty string - \in (epsilon)

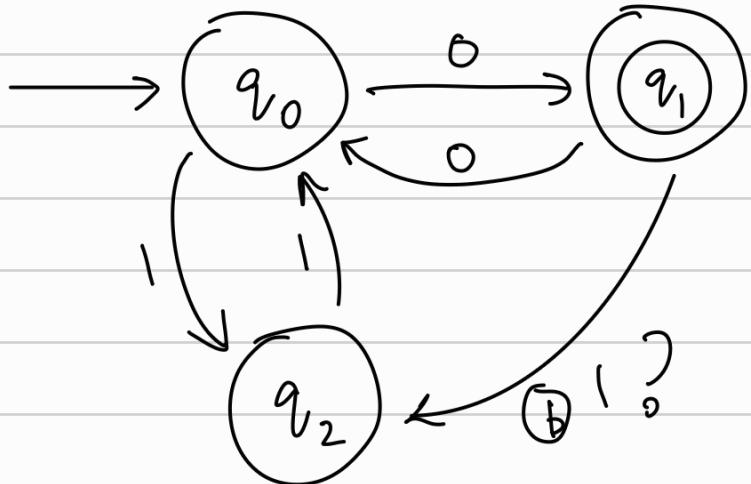
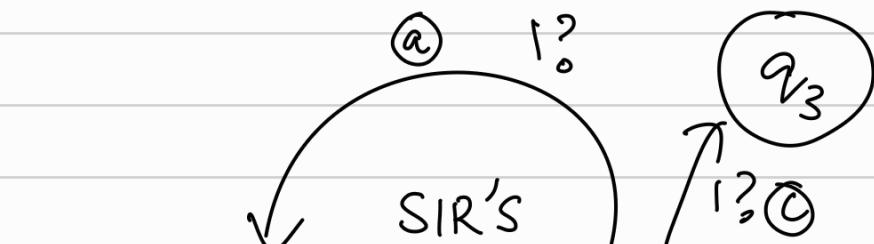
- in case of even no. of 1's

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

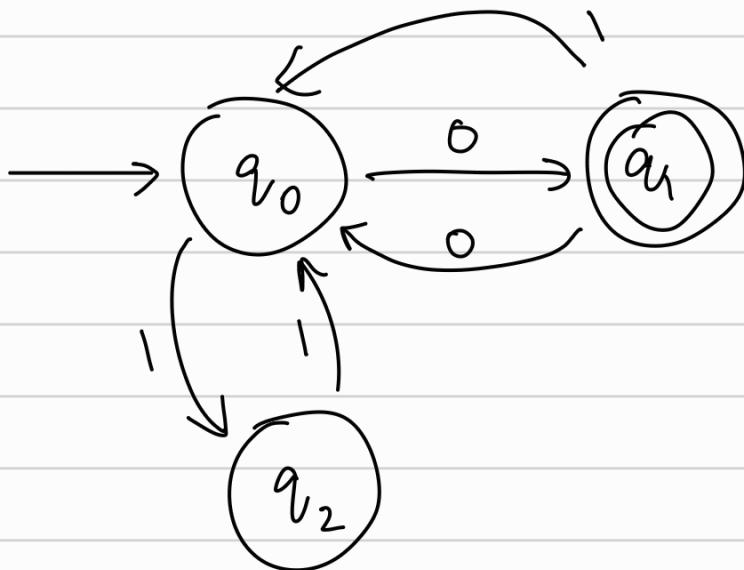


$\left\{ \begin{array}{l} \text{odd no of 0's,} \\ 1 \text{ is the least} \\ \text{size of string -} \end{array} \right\}$

$$\begin{aligned}\delta(q_0, 0) &= q_1 \\ \delta(q_0, 1) &= q_2 \\ \delta(q_1, 0) &= q_0 \\ \delta(q_1, 1) &= \end{aligned}$$

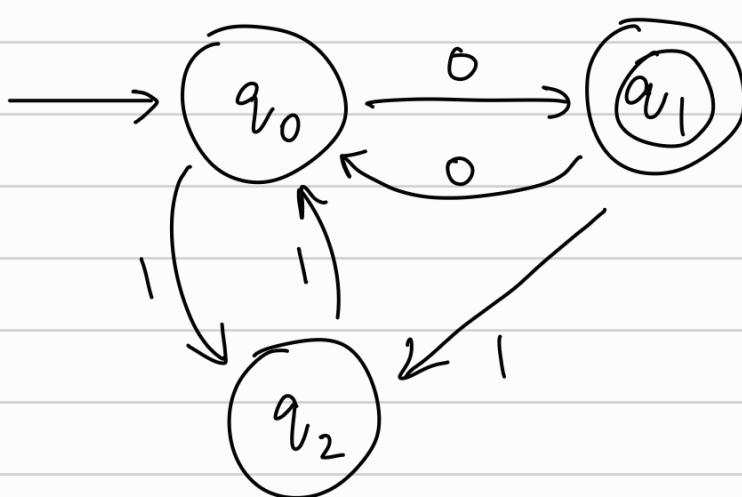


$$(a) \text{ let } \delta(q_1, 1) = q_0$$



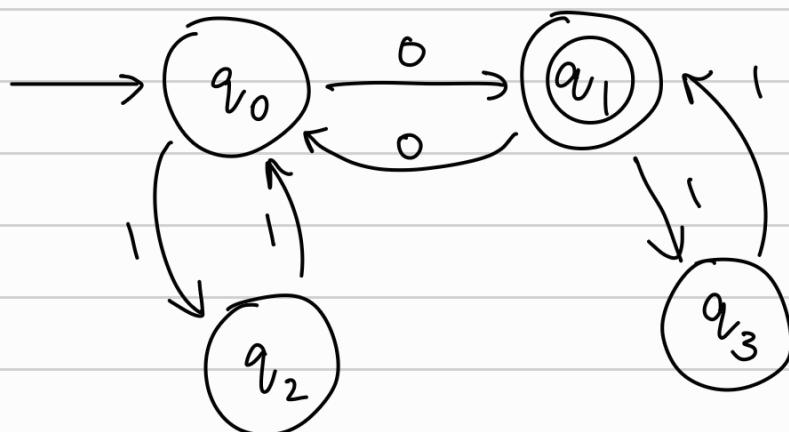
Machine accepts
010
↳ invalid.

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



Machine accepts
0110 → invalid.

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

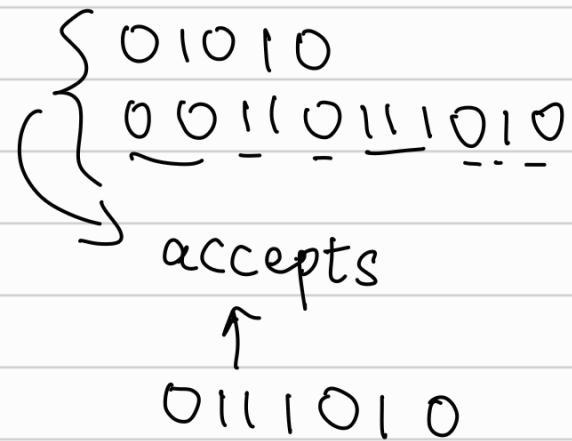


let $\delta(q_3, 1) = q_1$
 \therefore the loop
 $q_1 - q_3 - q_1$
tracks even 1's
like $q_0 - q_1 - q_2$

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) $q_2 \rightarrow$ seems right ...

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



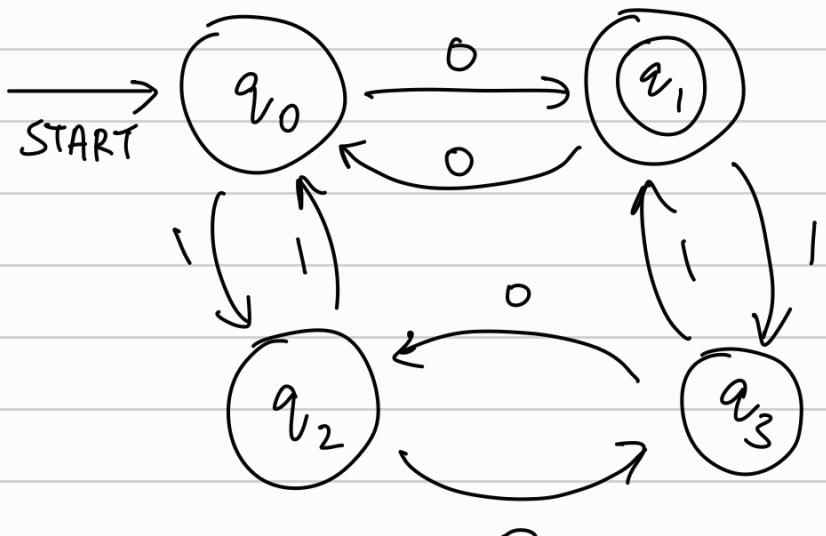
What abt $\delta(q_2, 0)$?

- 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

$$\begin{aligned}
 \delta(q_0, 0) &= q_1 - \\
 \delta(q_0, 1) &= q_2 -- \\
 \delta(q_1, 0) &= q_0 . \\
 \delta(q_1, 1) &= q_3 --- \\
 \delta(q_2, 0) &= q_3 --- \\
 \delta(q_2, 1) &= q_0 . \\
 \delta(q_3, 0) &= q_2 -- \\
 \delta(q_3, 1) &= q_1 -
 \end{aligned}$$



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 ???

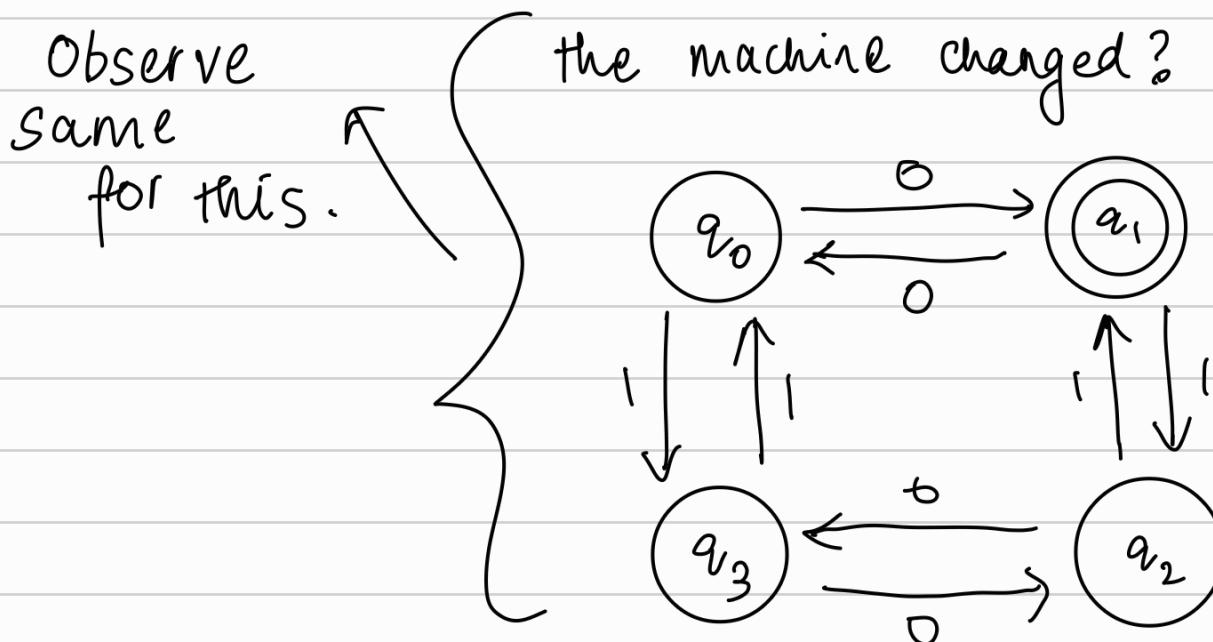
gain intuition.

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

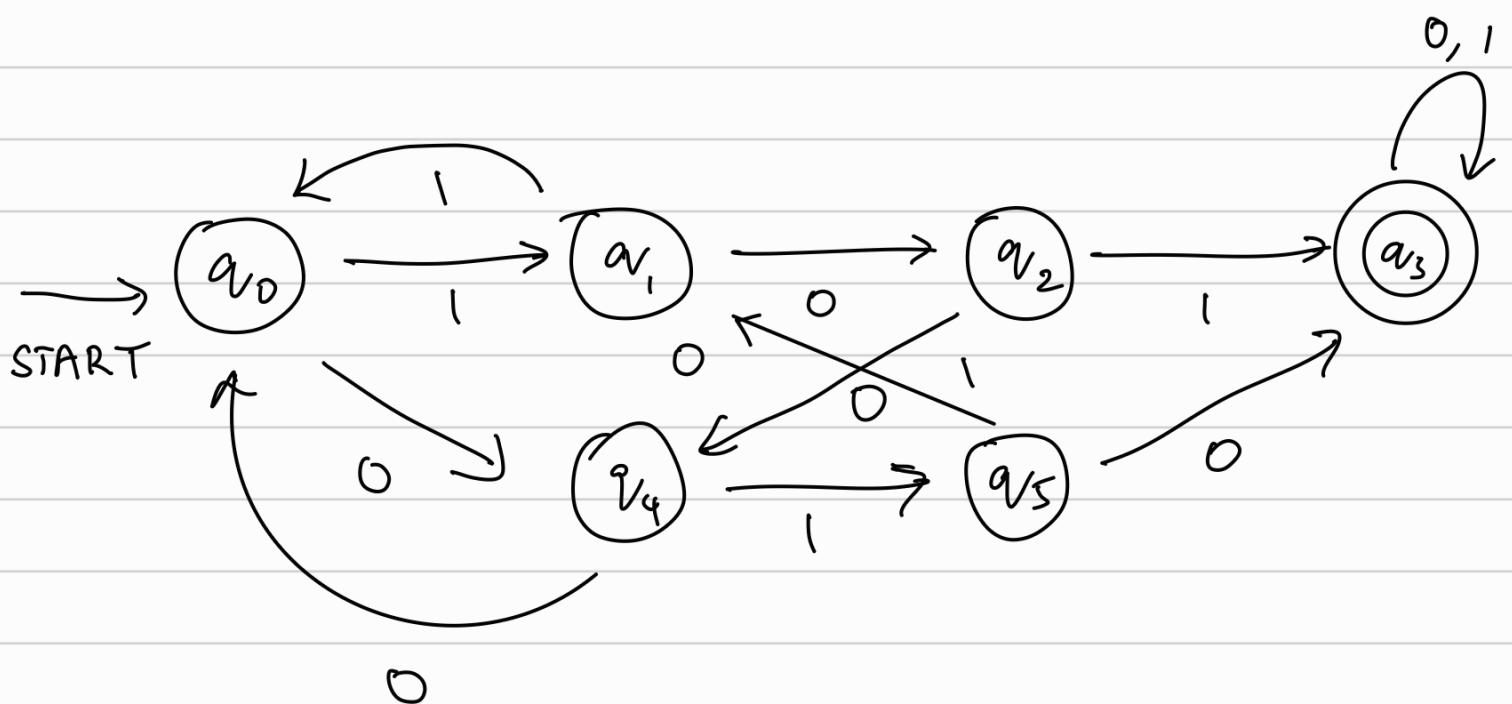
- (i) $\hat{\delta}(q_0, x) = q_0$ ||
(ii) $\hat{\delta}(q_0, x) = q_1$ ||
(iii) $\hat{\delta}(q_0, x) = q_2$ ||
(iv) $\hat{\delta}(q_0, x) = q_3$ ||

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.



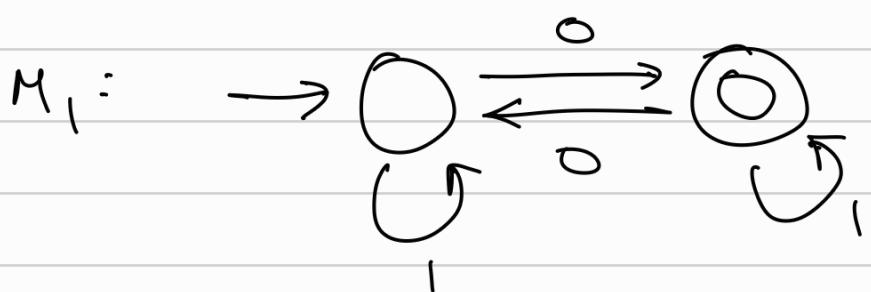
$$L = \left\{ x \in \{0,1\}^* \mid x \text{ contains } 101 \text{ or } 010 \text{ as Substring} \right\}$$



Given two FSAs M_1, M_2

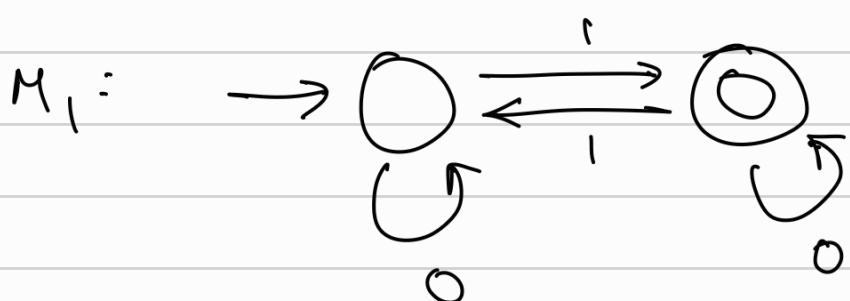
$$M_1 = \{Q, \Sigma, \delta_1, q_0^1, F_1\}$$

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



$M_1 \vee M_2$

(or)



$M_1 \wedge M_2$

PRODUCT AUTOMATON:

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

$q_0 \in Q_1 \times Q_2$ $f(F_1, F_2)$

$$\rightarrow (p_0, q_0) \quad p_0 q_1$$

$$p_1 q_0 \quad p_1 q_1$$

$$\delta((p_0, q_0), 0) = \delta(p_0, 0) \cup \delta(q_0, 0)$$

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

$$Q_1 = \{q_0, q_1\} ; Q_2 = \{p_0, p_1\}$$

$$Q_1 \times Q_2 = \{(q_0, p_0), (q_0, p_1), (q_1, p_0), (q_1, p_1)\}$$

$$\delta((p_0, q_0), 1) = (\delta_2(p_0, 1), \delta_1(p_0, 0)) = (p_1, q_0)$$

$$\delta((p_0, q_1), o) = \left(\underbrace{\delta_2(p_0, o)}_{\substack{\downarrow \\ \text{state of} \\ Q_2}}, \underbrace{\delta_1(q_1, o)}_{\substack{\downarrow \\ \text{state} \\ \text{of } Q_1}} \right) = (p_0, q_0)$$

$$M_1 \times M_2 = \left\{ Q_1 \times Q_2, \Sigma, \delta, (q_0, p_0), F \right\}$$

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

$$\textcircled{1} \quad \left\{ x \mid \begin{array}{l} \hat{\delta}_1(q_0, x) = q \in F_1 \\ \hat{\delta}_2(p_0, x) = q' \in F_2 \end{array} \right\}$$

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

$$\therefore M_1 \wedge M_2 = \left\{ Q_1 \times Q_2, \Sigma, \delta, (p_0, q_0), F_1 \times F_2 \right\}$$

$$\textcircled{2} \quad L = \left\{ x \mid \begin{array}{l} \hat{\delta}_1(q_0, x) = q \in F_1 \\ \text{or} \\ \hat{\delta}_2(p_0, x) = q' \in F_2 \end{array} \right\}$$

$$= \left\{ x \mid \begin{array}{l} \hat{\delta}((p_0, q_0), x) = (q, q'), q \in F_1 \\ \vee \\ q' \in F_2 \end{array} \right\}$$

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

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

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

using the above model, construct 010 \neq 101
q's.

$M_1 = 010$ as substring } \neq then do $M_1 \cup M_2$
 $M_2 = 101$ as substring } HOMEWORK.

given M_1, M_2 : how to construct $M_1^C \neq M_2^C$.

\downarrow \downarrow
 x doesn't contain y x doesn't contain y'

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

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

$$q_f^1 \in F_1, q_f^2 \in F_2. \quad \left(\begin{matrix} q_f^1 \\ q_f^2 \end{matrix} \right) \xrightarrow{\quad} x \in L(M_1 \wedge M_2)$$

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

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

$$x = (q_{nf}^1, q_f^2) \rightarrow x \in L(m_2/m_1)$$

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

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

$\Downarrow a \in \Sigma$

$$\hat{\delta} \left((q_0^1, q_0^2), x \right) = \underbrace{?}_{\substack{\hookrightarrow x \in \Sigma^*}} \longrightarrow \text{based on machine} \dots$$

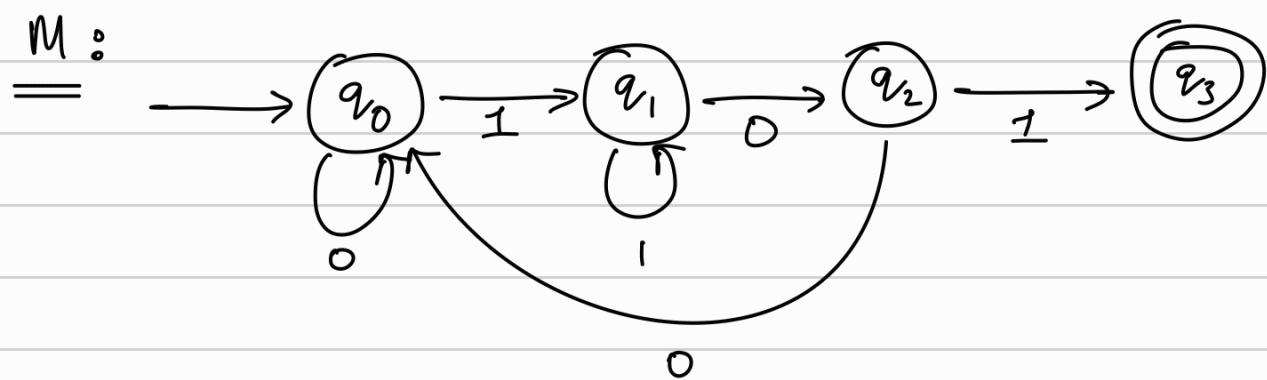
$$\begin{aligned}
 F(M_1 \wedge M_2) &\rightarrow M_1 \wedge M_2 \rightarrow F = F_1 \times F_2 \\
 &\quad (\text{both machines accept}) \\
 F(M_1 \vee M_2) &\rightarrow M_1 \vee M_2 \rightarrow F = (F_1 \times Q_2) \vee (Q_1 \times F_2) \\
 &\quad (\text{either of them are true}) \\
 F(M_1 \setminus M_2) &\rightarrow (q_f^1, q_{nf}^2) \rightarrow F = F_1 \times F_2^c \\
 &= (F_1 \times \underline{\underline{(Q_2 \setminus F_2)}}) \\
 F(M_2 \setminus M_1) &\rightarrow F = F_1^c \times F_2 \\
 &= ((Q_1 \setminus F_1) \times F_2) \\
 &\quad \hookrightarrow \text{non-final states.}
 \end{aligned}$$

(e.g.) $\hat{s}\left((q_0^1, q_0^2), x\right) = (q, q') \in F_1^c \times F_2$
 $\Rightarrow x \in L(M_2 \setminus M_1)$.

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

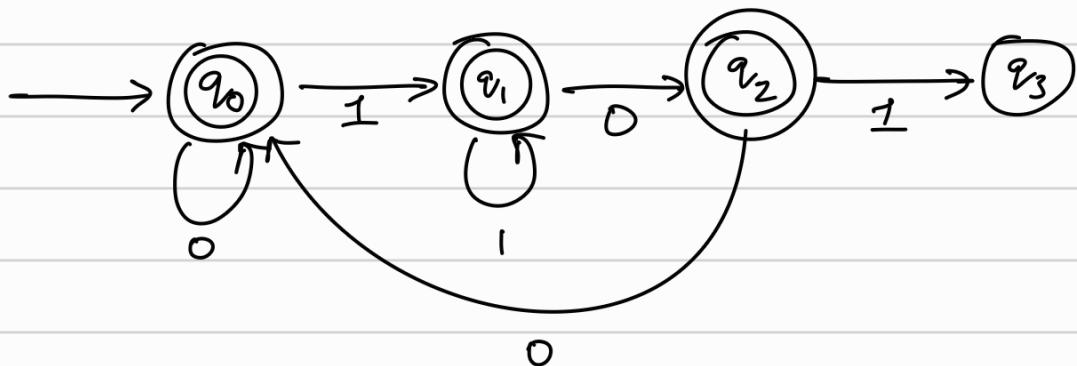
$$L_1 = \left\{ x \mid x \text{ doesn't contain } 101 \text{ as substring} \right\}$$

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

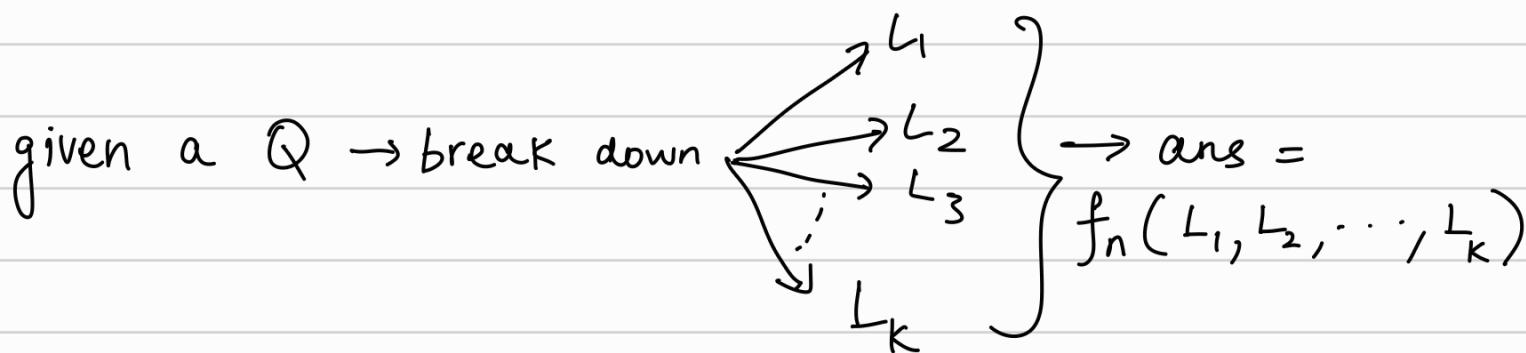


$$L(M_1) = L(M^c)$$

M₁ (doesn't contain 101) :



"accept what's not by M, reject what's accepted by M"



(e.g.) $L = \left\{ x \mid \begin{array}{l} \text{x begins with 00} \rightarrow L_1 \\ \text{x ends with 11} \rightarrow L_2 \\ \text{x not containing 1010} \rightarrow L_3^c \end{array} \right\}$

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

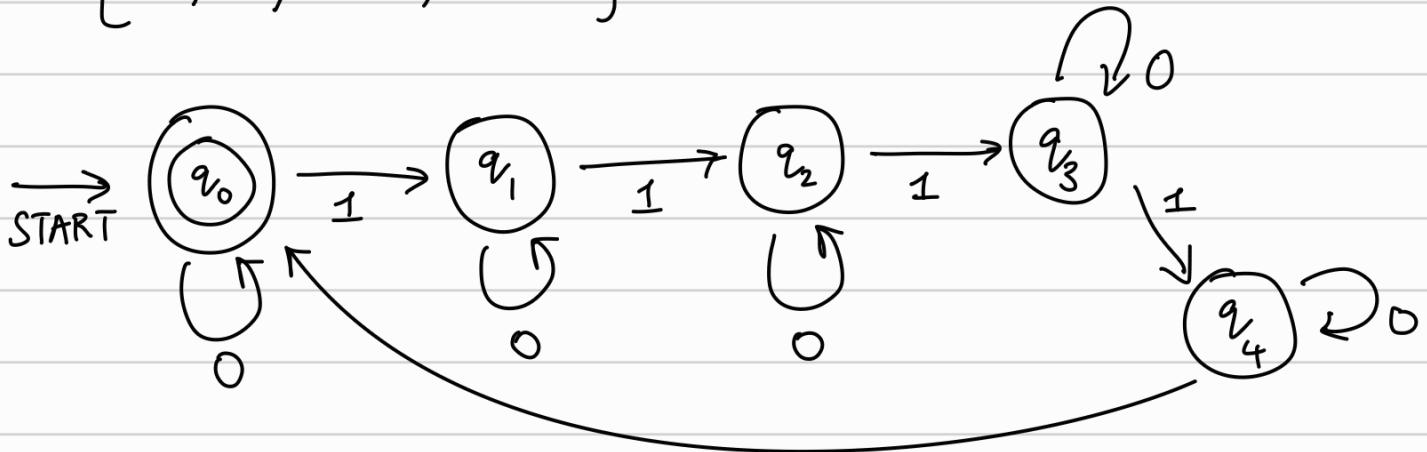
$$L_2 = \left\{ x \mid x \text{ contains } \frac{101}{P_1} \text{ & doesn't contain } \frac{11}{P_2} \right\}$$

HOMEWORK.

P.T.O.

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

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

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


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

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

usually $\delta: Q \times \Sigma \rightarrow Q$, 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 ok. where the machine will search for next state. such machines are "Non-Deterministic Finite Automaton".

Finite Automatons (FA)

Deterministic FA (DFA)

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

Non-Deterministic FA (NFA)

$$\delta: Q \times \Sigma \rightarrow 2^Q$$

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 .

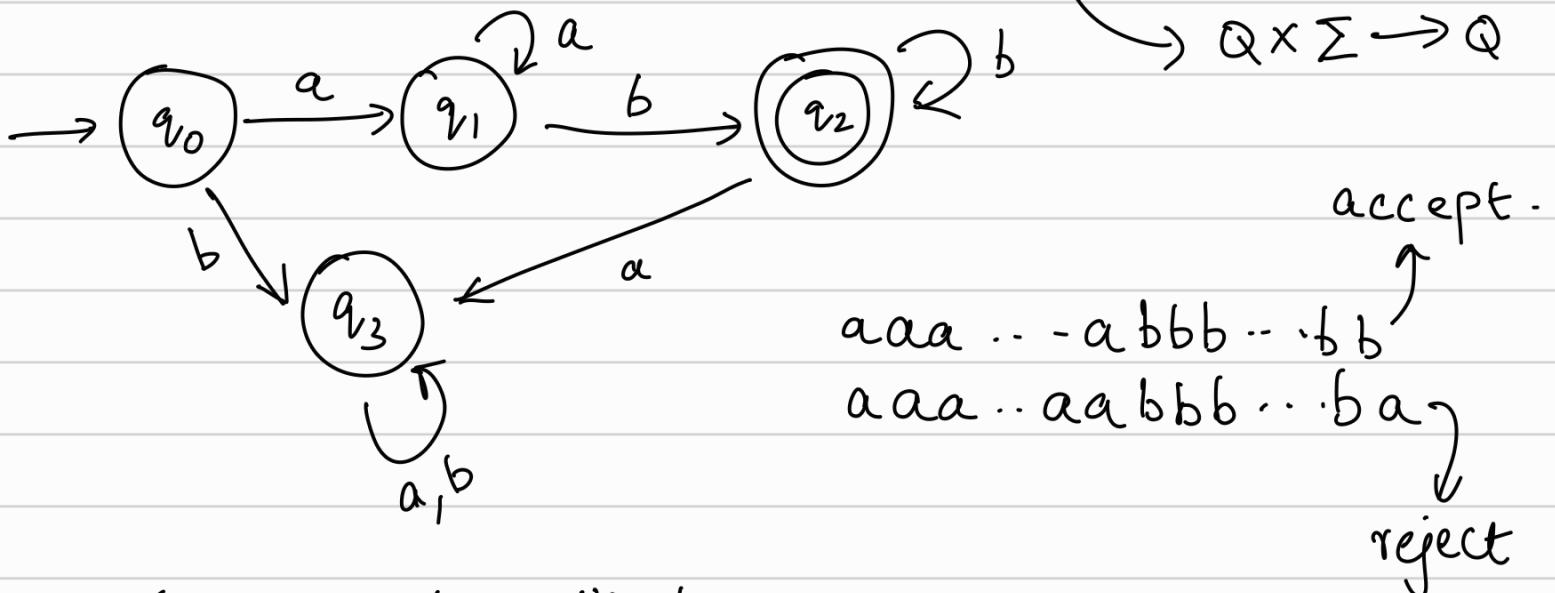
By observation,

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

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

P.T.O.

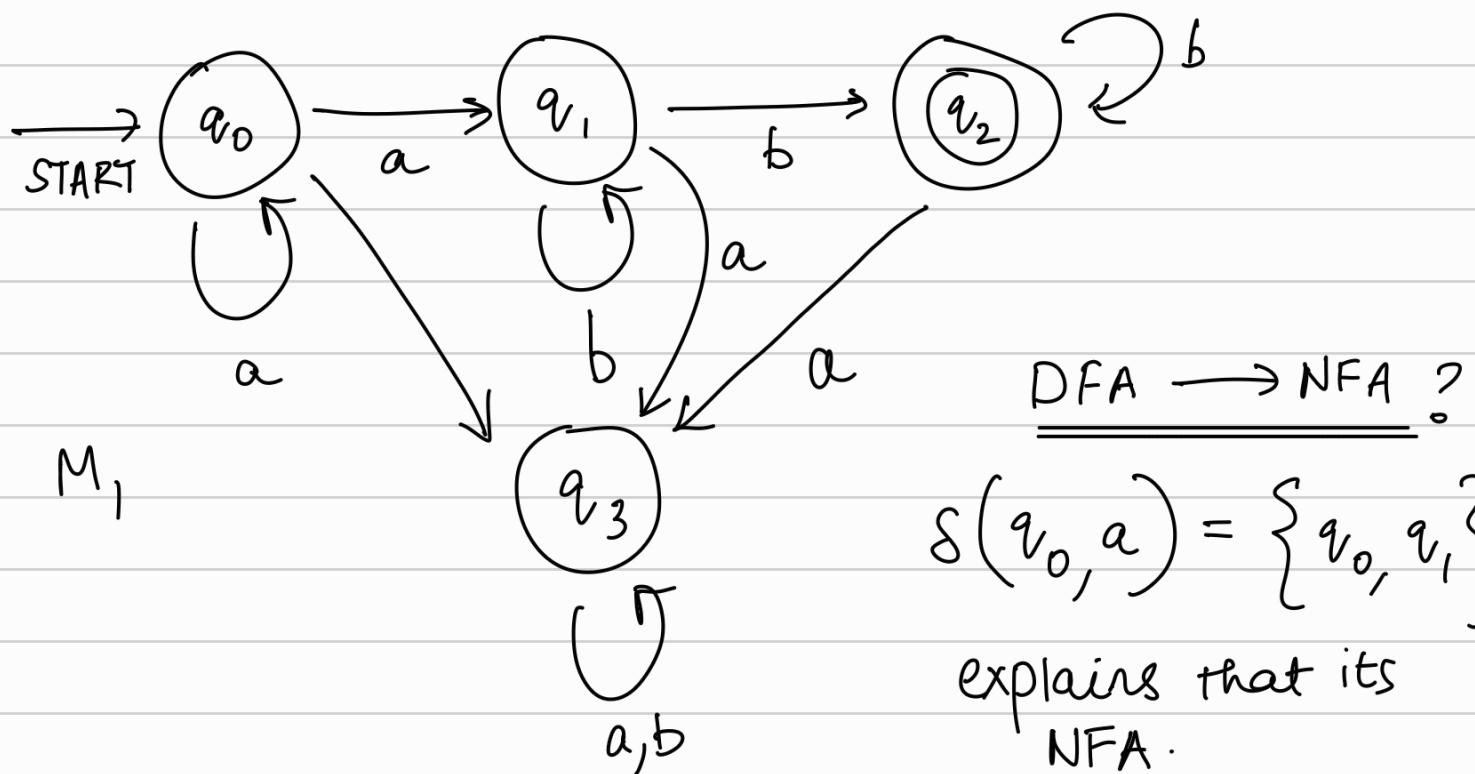
$$M = \left\{ Q = \{q_0, q_1, q_2, q_3\}, \Sigma = \{a, b\}, \delta, q_0, F = \{q_2\} \right\}$$



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

language accepted

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



$M_1 \rightarrow$ accepts same language as M .

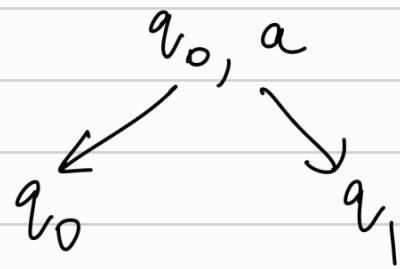
(e.g.) $a^3 b^2$

$q_0 q_0 q_1 q_1 q_2$ shall work.

$q_0 q_0 q_1 q_2 q_2$ shall also work.

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

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



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

for DFA \rightarrow the graph is a path.

Class P \rightarrow \exists det. poly time algo \rightarrow \exists DFA
 Class NP \rightarrow \exists non-det poly-time algo \rightarrow \exists NFA.

P = NP \rightarrow if \exists NFA \exists 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?

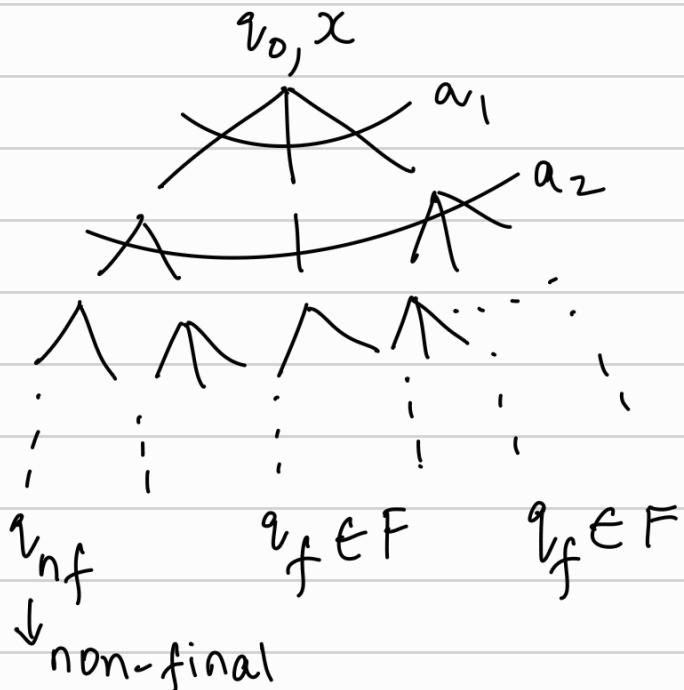
~~DFA~~

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

$$\hat{\delta}(q_0, x = a_1 a_2 \dots a_n) = \underbrace{\left(\left(\left(\left((q_0, a_1), q_2 \right), a_3 \right), q_4 \dots \right), a_n \right)}_{a_i \in \Sigma}$$

$\hookrightarrow \exists$ a unique path.

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



If $\hat{\delta}(q_0, x) = q_f \in F$, 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)

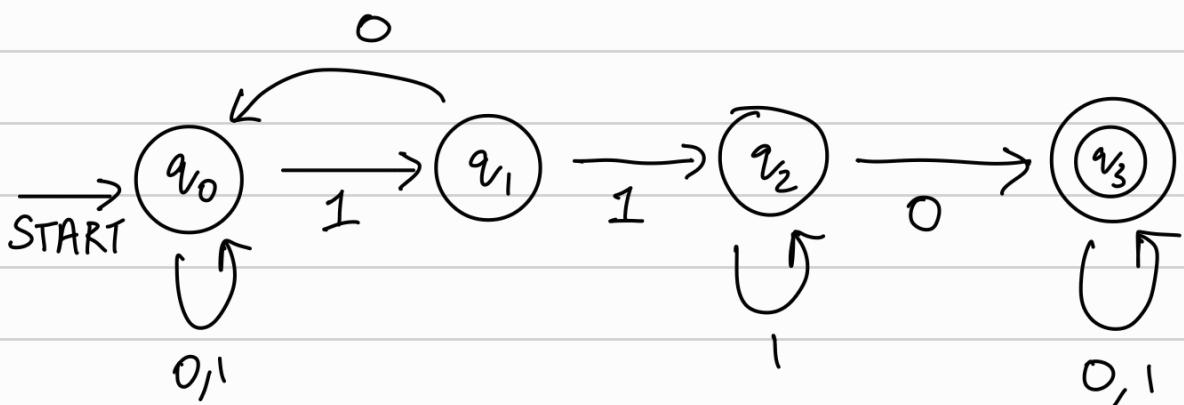
- if $x \in L(M)$, then NFA will INDEED stop at q_f ; SOMEHOW discovers path from q_0 to q_f ...
- hypothetical only...
- can simultaneously simulate $x @ q_0$ (or) parallelly explore ALL paths...
- not AI as AI has inaccuracies...
- does introducing NFA increase the computing power of the automaton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

(e.g.)

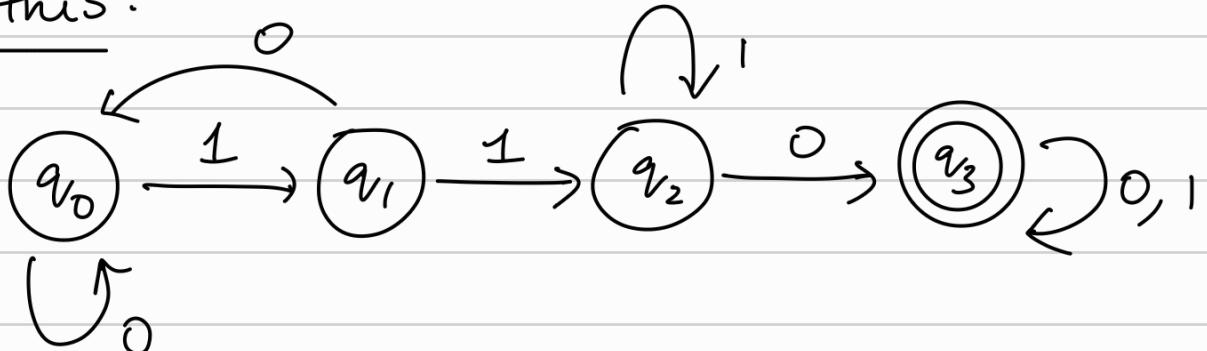
$$L = \{x \mid x \text{ contains } 110 \text{ as substring}\}$$

NFA for this:

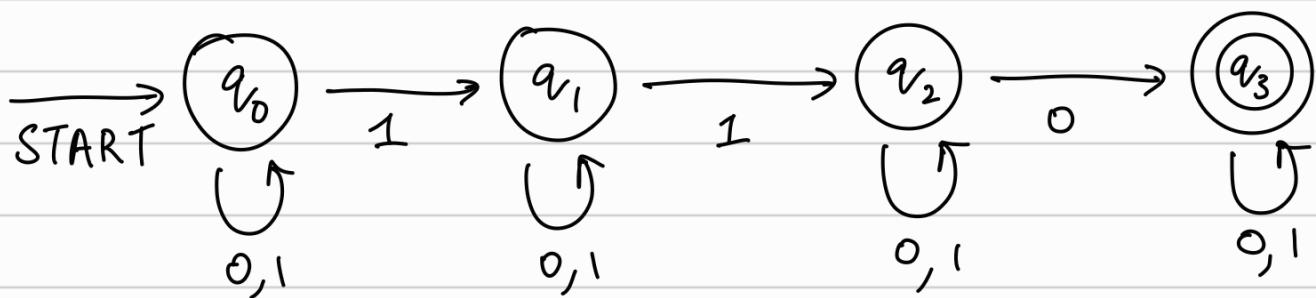
$\forall x \in L(M) \exists \text{ path } q_0 \dots q_3$



DFA for this:



Another NFA proposed :



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

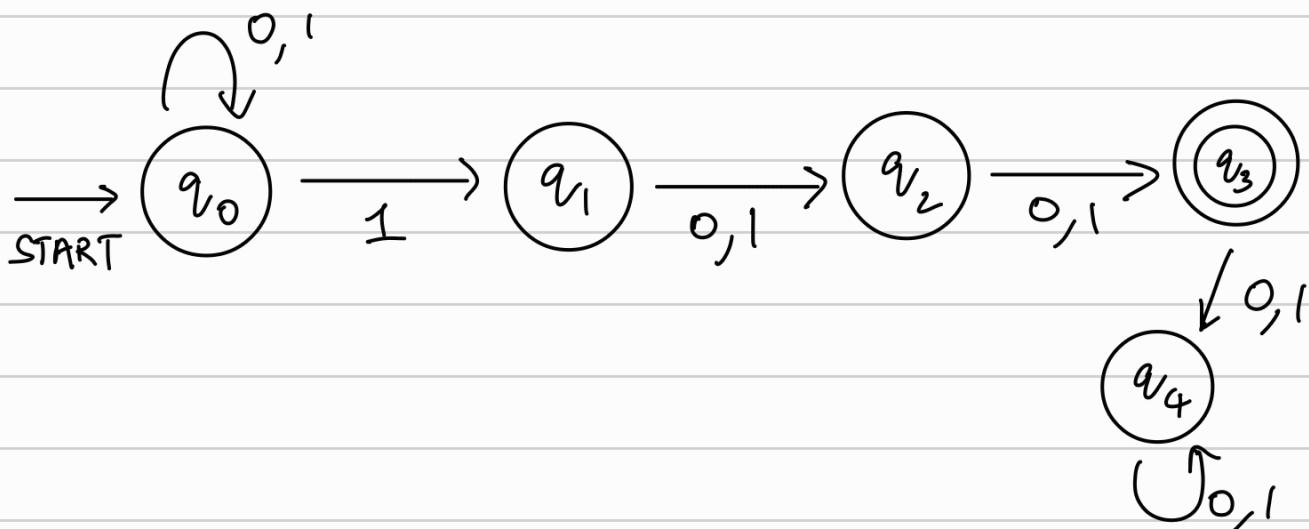
for FA: if valid str \rightarrow ↗ path $q_0 \dots q_f //$
invalid str \rightarrow ↗ path $q_0 \dots q_{nf} //$

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

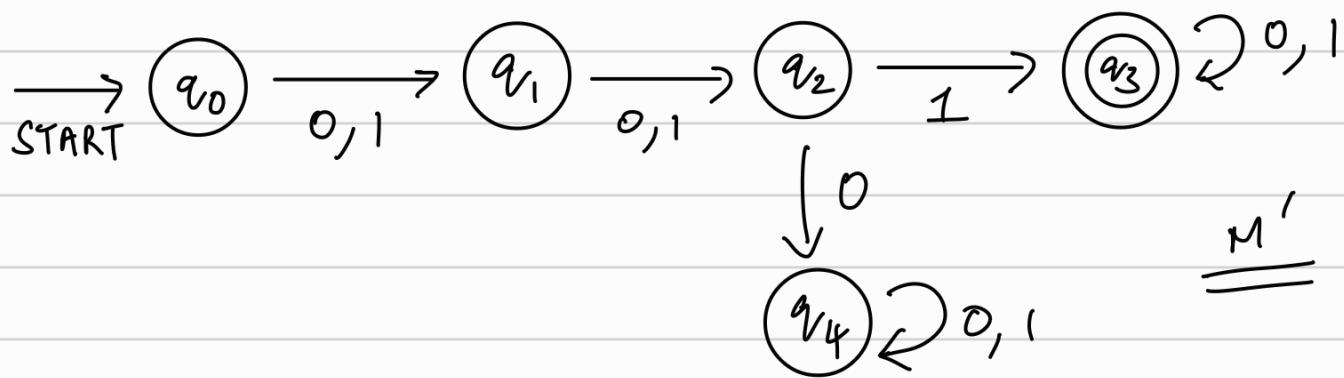
$$L = \{101, 000101, 10100110, 000100, \dots \dots \}$$

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

does ↗ an FA for this?

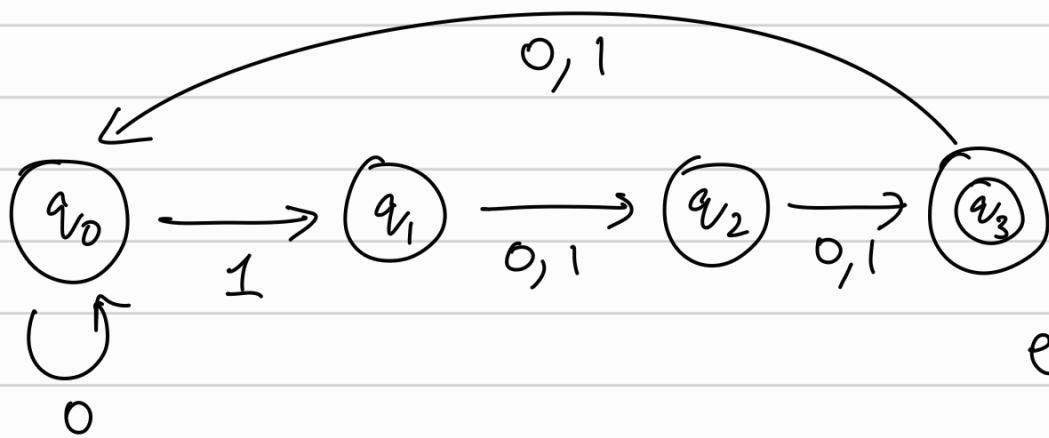


what if we reverse the string & find 3rd from left?



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

Also,

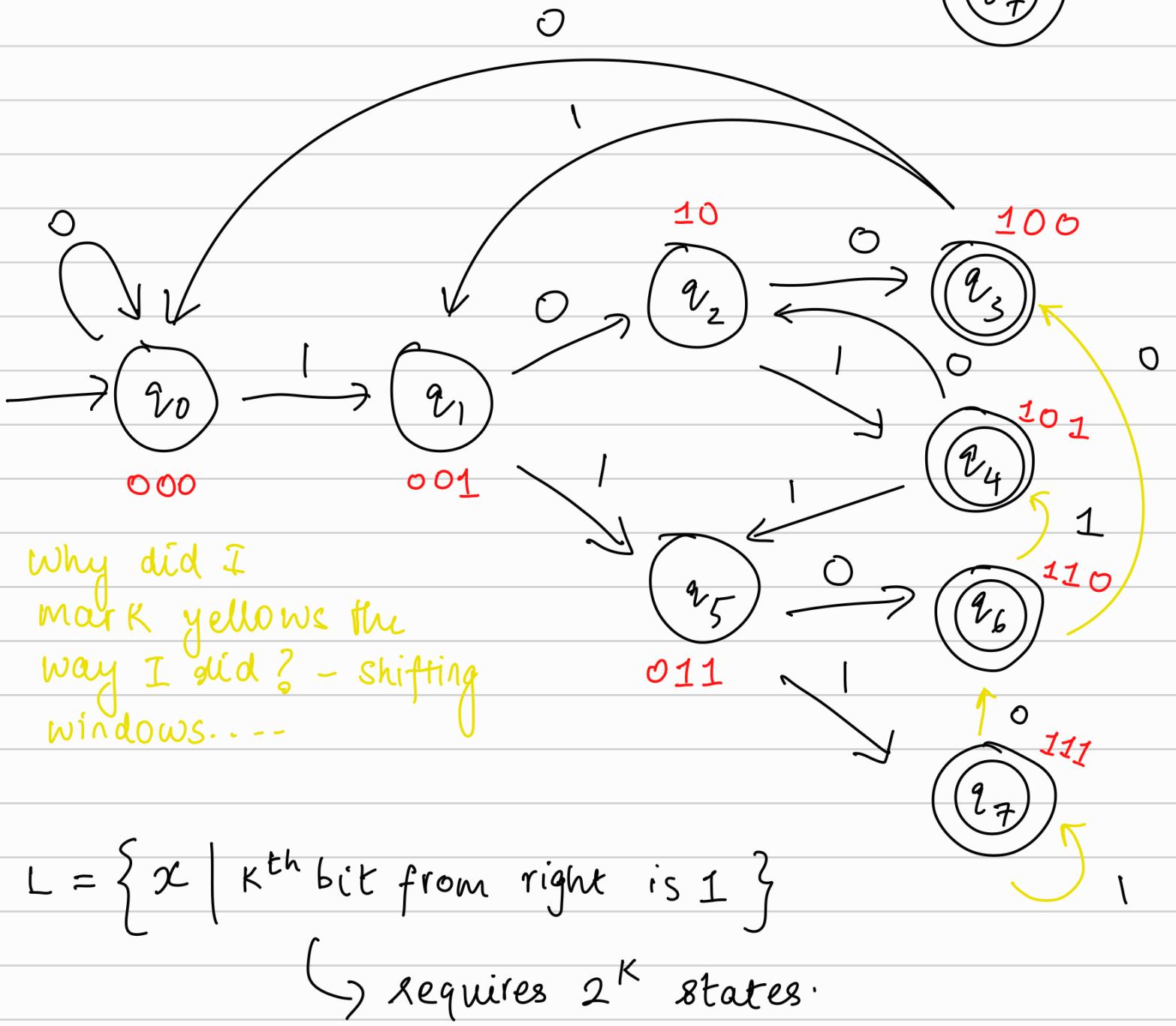
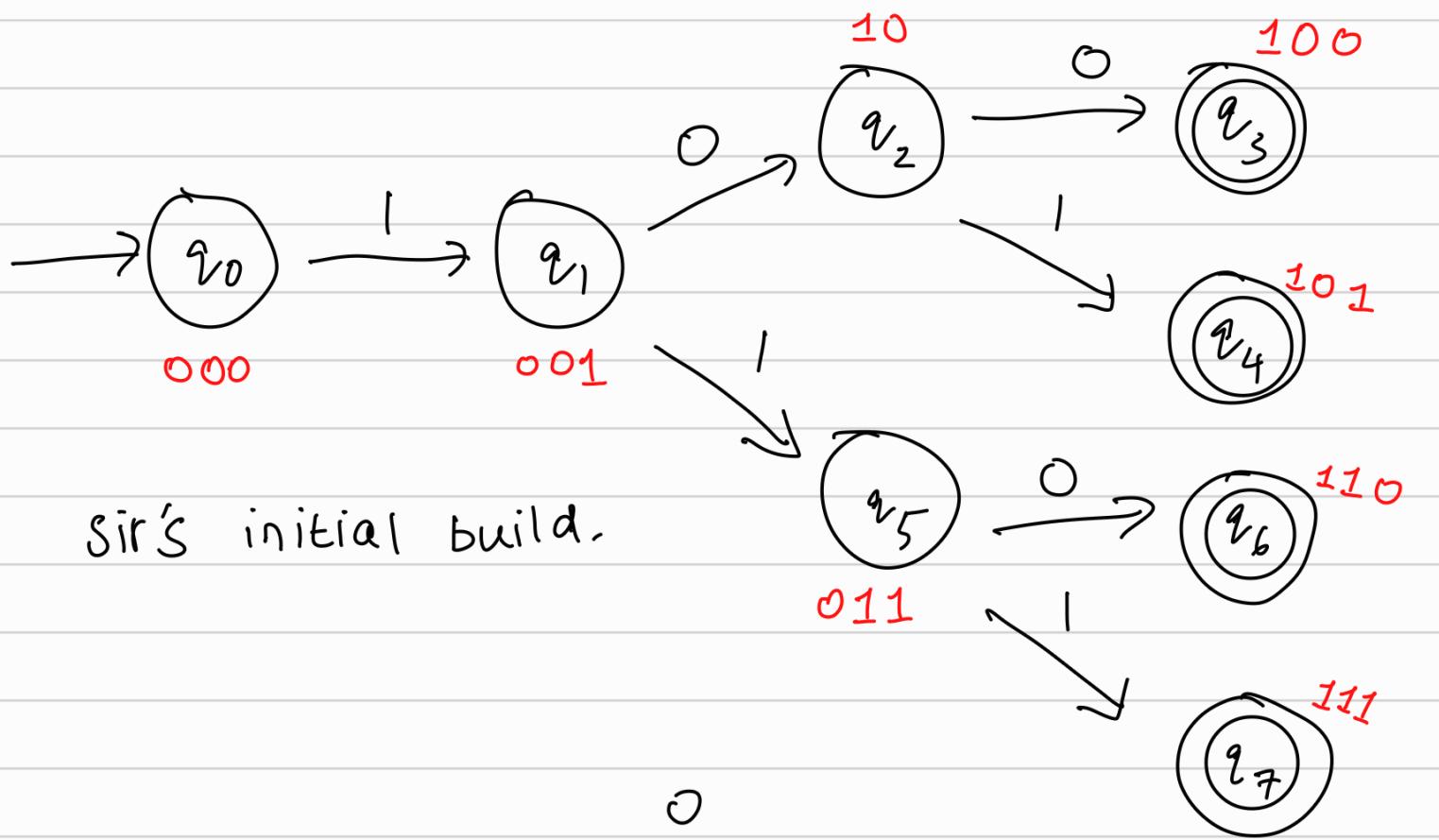


exists...

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.



REGULAR EXPRESSIONS (RE)

{rep. ∞ string in finite form}

* Algebraic representation.

* Expresses computing power of FA.

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

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

zero or
more 0's.

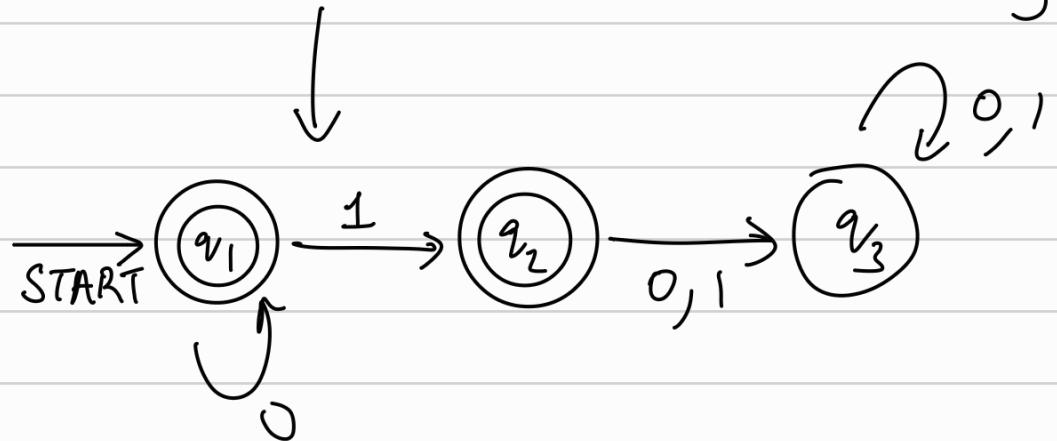
RE_1
0

RE_2
1

let $RE_3 = 0 + 1$.

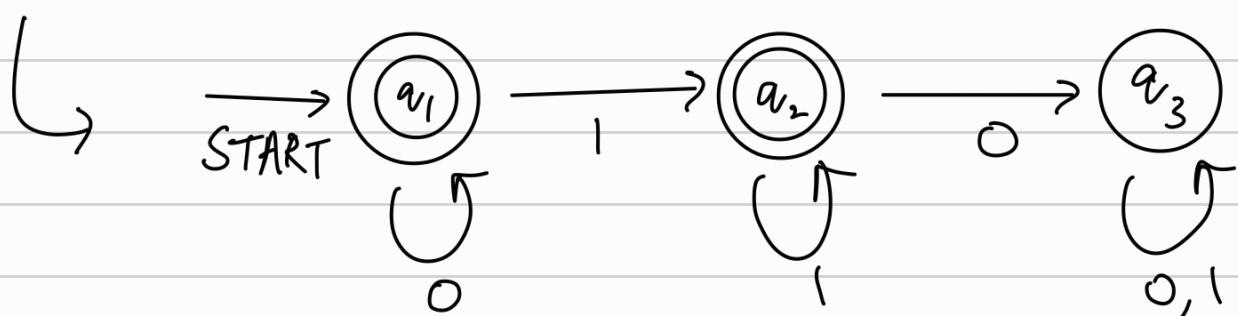
$\therefore L(RE_3) = \{0, 1\}$

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



$0^* + 1^*$

THIS IS WRONG. WHY?

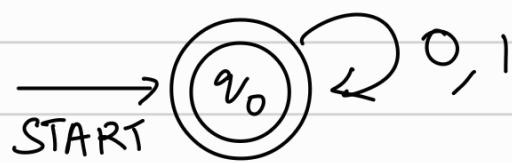


$(0+1)^*$ → zero or more copies of $(0+1)$

$\left\{ \epsilon, (0+1), (0+1)(0+1), (0+1)(0+1)(0+1), \dots \right\}$

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
...

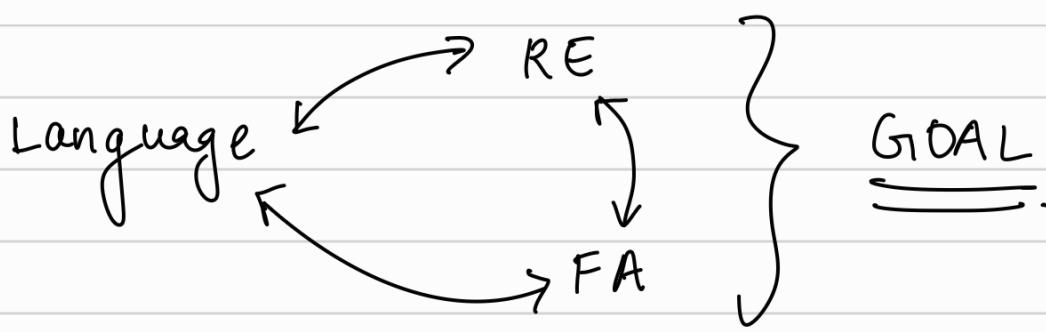
$(0^* + 1^*)^*$ → my thought: isn't this same as prev?

$\left\{ \epsilon, 0^* + 1^*, (0^* + 1^*)(0^* + 1^*), (0^* + 1^*)(0^* + 1^*)(0^* + 1^*), \dots \right\}$

∴ $(0+1)^* \equiv (0^* + 1^*)^*$

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

$L_1 = (0+1)^*$ 101 $(0+1)^*$
RE = $\overbrace{RE_1}^{\downarrow} \quad RE_{2,3,4} \quad \overbrace{RE_5}^{\downarrow} \quad //$



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

$$RE(L_2) = \underbrace{(0+1)^*}_{RE_1} \underbrace{\underline{1}}_{RE_2} \underbrace{(0+1)}_{RE_3} \underbrace{(0+1)}_{RE_4}.$$

3rd from
right
 .

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

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

↓

$\hookrightarrow 0 \dots 0$
is a valid string.

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

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

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

$$\begin{aligned} \text{DFA} &\rightarrow \Sigma = \{0, 1\}, \delta: Q \times \Sigma \rightarrow Q \\ \text{NFA} &\rightarrow \Sigma = \{0, 1\}, \delta: Q \times \Sigma \rightarrow 2^Q \end{aligned}$$



can NFA move read control without reading i/p symbol

↳ does this increase computational power of FA.

$$010 \in \epsilon 0 \epsilon \epsilon \dots \epsilon 10$$

$$\text{for a symbol } a, \epsilon^* a = a \epsilon^* = \epsilon^* a \epsilon^* = a$$

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

↓
empty

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

if $\epsilon \rightarrow$ empty string, then use $\overset{n}{\underset{\epsilon}{\delta}}(q_0, \epsilon)$
 → empty symbol, then use $\delta(q_0, \epsilon)$

does \exists DFA for each ϵ -NFA?

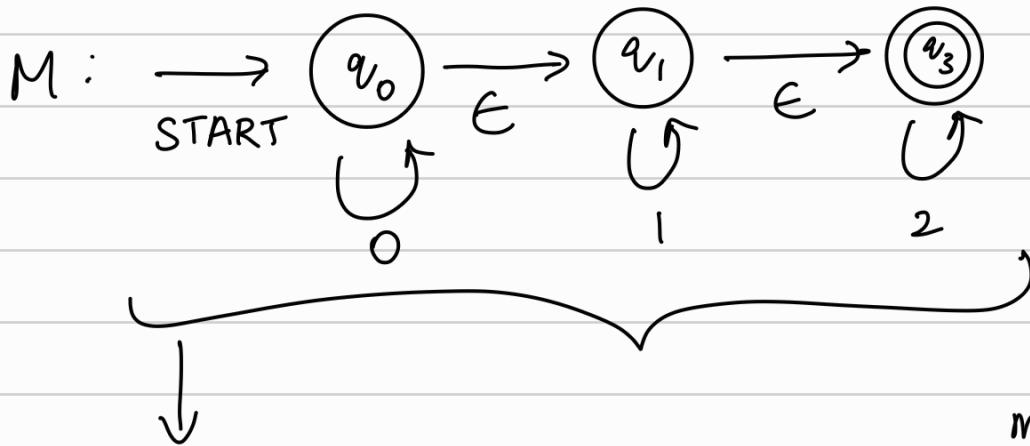
(basically ploncking ϵ between any 2 consecutive symbols to possibly achieve something greater??)

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

for $a \in \Sigma$, $\delta^D(q, a) \rightarrow$ deterministic (no uncertainty)
 $\delta^N(q, a) \rightarrow$ uncertainty $\rightarrow \epsilon^* a \epsilon^*$

$L = (0+1)^*$ · does \exists 2 DFA's?
if not, P.T.
 \nexists DFA.

\downarrow
be anything
 $\epsilon a \epsilon \epsilon$
 $\epsilon \epsilon a \epsilon$
 a
 $\epsilon \epsilon \epsilon a \dots$
+ so on...



$$\epsilon \epsilon \epsilon = \epsilon$$

$$0 \epsilon \epsilon = 0$$

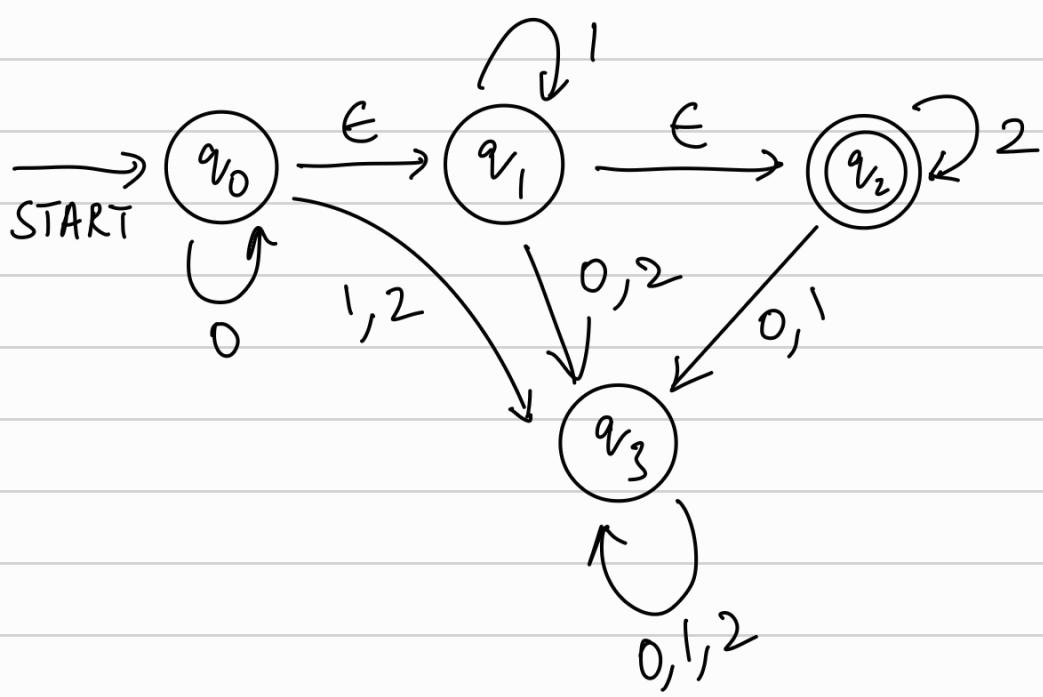
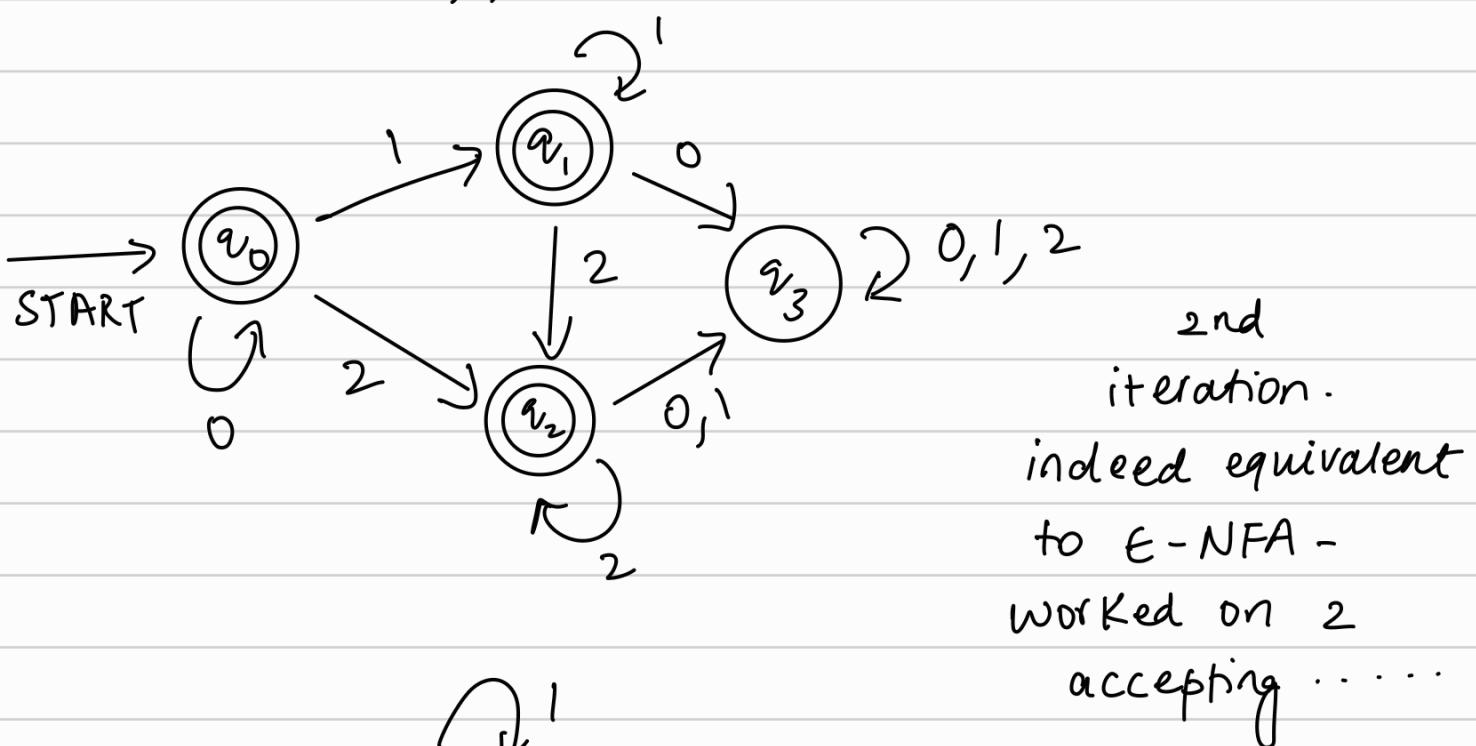
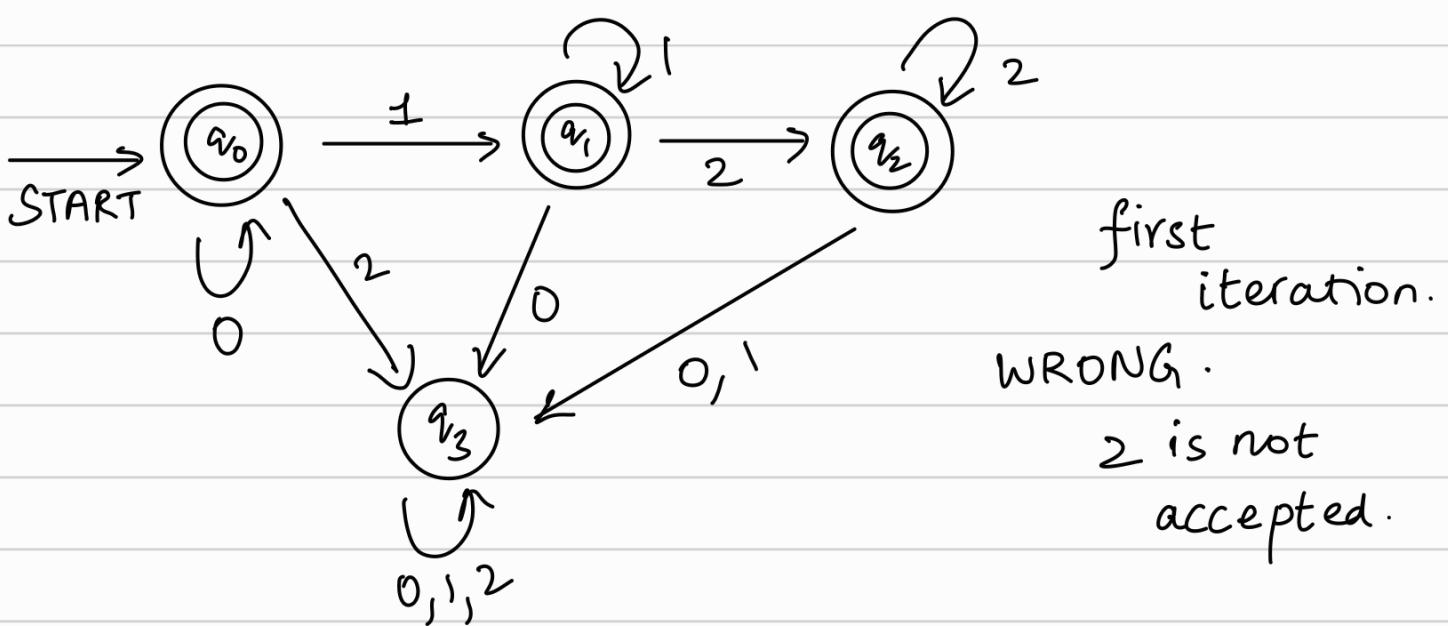
$$\epsilon 1 \epsilon = 1$$

$$\epsilon \epsilon 2 = 2$$

$$0 0 \epsilon 1 \epsilon 2 = 0 0 | 2$$

$$0 0 \epsilon 2 2 2 = 0 0 2 2 2$$

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



Epsilon Closure:

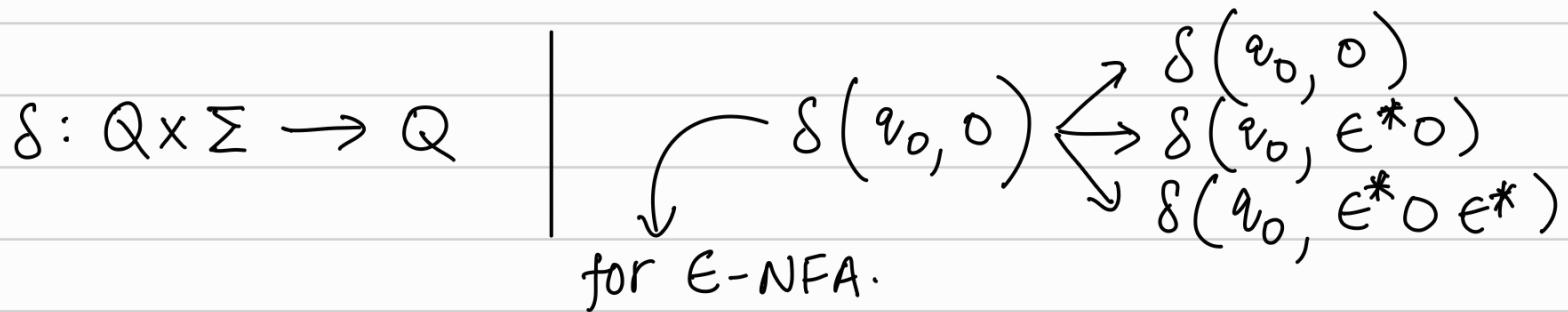
ϵ -closure of q is the set of all q' 's.t.
 $q' \in Q$ and q' is reachable from q by reading
one/more ϵ 's.

$\forall q \in Q, q \in \epsilon\text{-closure of } q, E(q)$

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

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

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



$$L = \left\{ x \in \{0,1\}^* \mid \begin{array}{l} \#_0(x) \text{ : ODD} \\ \#_1(x) \text{ : EVEN} \end{array} \right\} \xrightarrow{\text{think of RE for this}}$$