Chomsky hierarchy of Languages

Classes of languages

Machine Models

meaning: type of machines

that are capable of representing these languages

1. Regular languages → DFA, NFA → Finite state automata
2. CFG's → PDA → FSA + Stack
3. Recursive languages → Turing Machines → FSA + one side ∞ tape

more elements allowed in machine models increasing

⇓

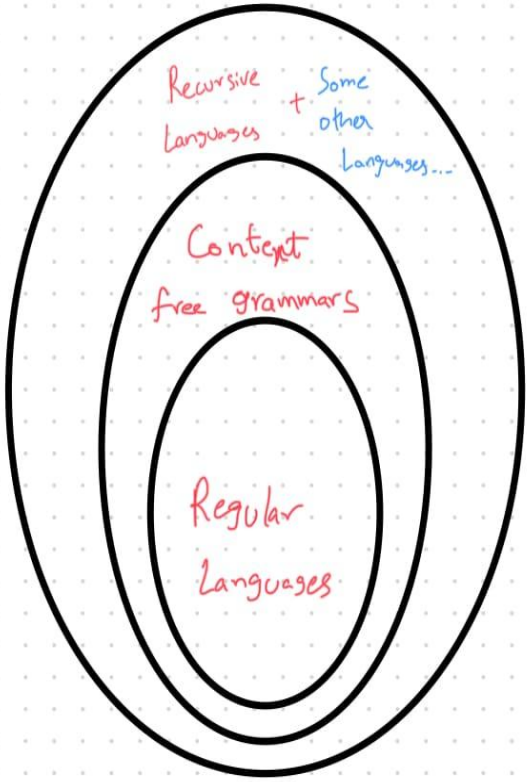
Machine Models can represent more languages.

Regular Languages

→ Lowest level languages in chomsky hierarchy.

→ We call a language regular if we are able to construct a DFA (or) NFA (or) write a Regular expression for that language.

⟶ Simply, if there exists a Finite state Machine for it, then it is a Regular Language



DFA (Deterministic Finite Automata)

→ It is a Machine Model we can use to construct machines for regular Languages

where L is a Regular Language.

→ Every DFA is represented by following things,

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

where $Q \rightarrow$ Set of states

$\Sigma \rightarrow$ alphabet set

$q_0 \in Q \rightarrow$ start state (always single)

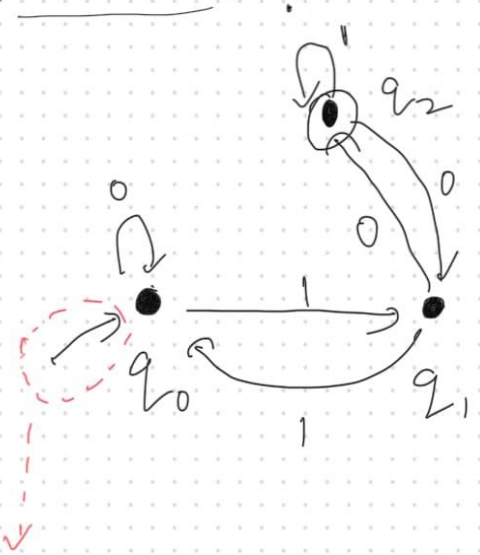
$F \subseteq Q \rightarrow$ set of final states (can be multiple)

$\delta : Q \times \Sigma \rightarrow Q \rightarrow$ Transition function.

→ Again, this is a machine model, we construct different Machines for different Languages.

ex:- Consider the following DFA that represents some arbitrary language. (L_1). (will see how to use this in a few minutes)

transition diagram of DFA of L_1 :-



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

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

$\rightarrow q_0 = q_0, \; F = \{q_2\}$

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

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

$\delta(q_0, 1) = q_1 \quad \delta(q_1, 1) = q_0$

$\delta(q_2, 0) = q_1$

$\delta(q_2, 1) = q_2$

In transition diagram,

→ We represent states using dots (or) circles.

→ we identify the single start state using an arrow pointing to it.

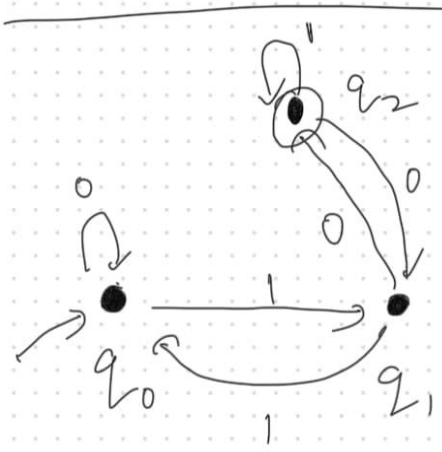
→ Final states are all circled around them (like q_2)

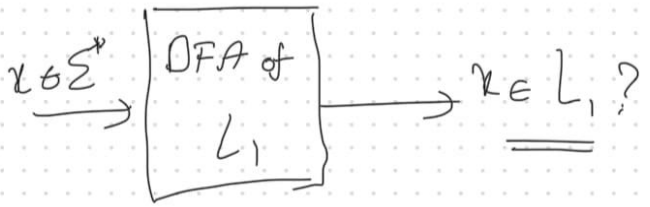
→ For every state, transitions to different (or) same state must be mentioned for every symbol in Σ.

How to use a given DFA :-

Same state must be mentioned for every Symbol in Σ.

How to use a given DFA :-





(i) $x_1 =$ "101"

$q_0 \to q_1 \to q_2 \to q_2 \in F \Rightarrow x_1 \in L_1$

(ii) $x_2 =$ "10110"

$q_0 \to q_1 \to q_2 \to q_2 \to q_2 \to q_1 \to q_0 \notin F \Rightarrow x_2 \notin L_1$

→ Start with start state.

→ Read characters in given string x one-by-one and keep transitioning between states guided by character you see in x and transition rules for current state.

→ After reading whole x, if you end up in a state "q" that is one of final states, then $x \in L$.

$$q \in F \to x \in L$$

Transition Table and Transition Diagram

↓ a table to represent our transitions

↳ Diagrammatic representation of our DFA.

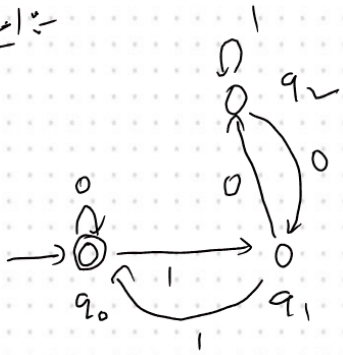
$DFA = \{ S, \Sigma, q_0, F, \delta \}$

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

Examples :-

Consider given DFA's below representing some language and then look at the transition table.

ex-1 :-



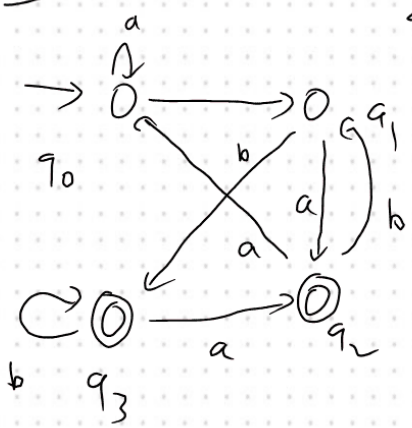
$\Sigma = \{0, 1\}$

Transition table for ex-1 →

	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

* For DFA machines, δ will output always a single state for a given (q_i, a), $q_i \in Q$, $a \in \Sigma$. So, write those all transitions in a table → transition table.

ex-2 :-



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

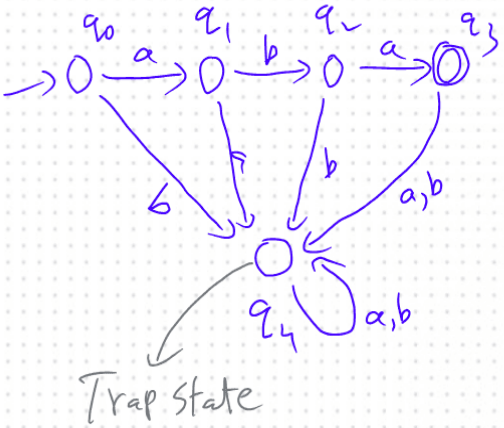
Transition table →

	a	b
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_0	q_1
q_3	q_2	q_3

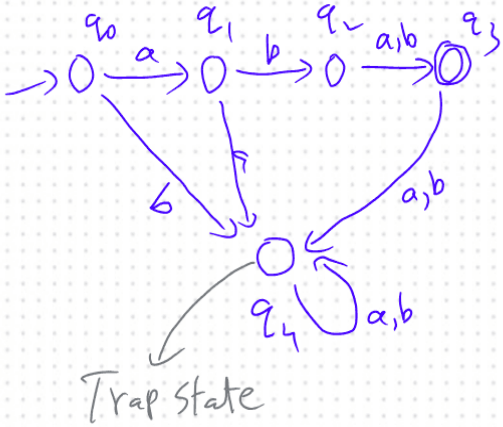
Constructing DFA for different Languages

examples :-

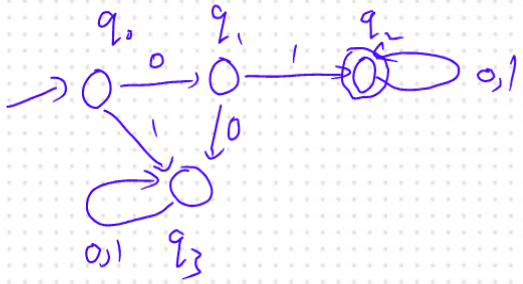
$L_1 = \{"aba"\}, \Sigma = \{a, b\}$



$L_2 = \{"aba", "abb"\}, \Sigma = \{a, b\}$

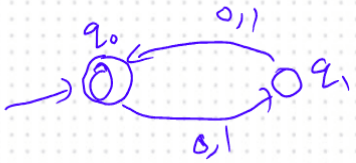


$L_3 = \{x \mid x \text{ starts with "01"}\}, \Sigma = \{0, 1\}$

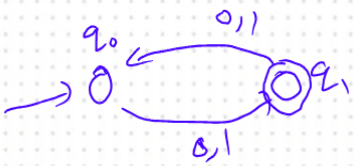


→ These DFAs for L_1, L_2, L_3 are constructed in such a way that they only reach final states for the strings in Language. Rest all strings always end in trap state.

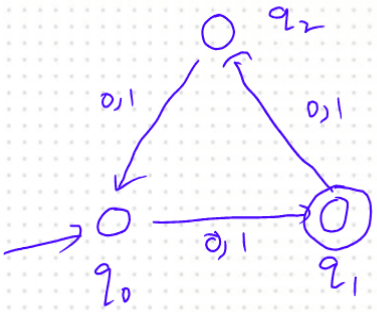
$L_4 = \{x \mid |x| \text{ is even}\}$ ← length of 'x' string



$L_5 = \{x \mid |x| \text{ is odd}\}$



$L_6 = \{x \mid |x| = 3n+1, n \geq 0, n \in N\}$



A DFA is a machine where only strings in your language will end in a final state and strings not in L are rejected.

Look at these DFAs and try to understand and develop reasoning on why they accept only strings in given L.

$L_7 = \{x \mid x \text{ has last } 2^{nd} \text{ char as '1'}\}, \Sigma = \{0, 1\}$

→ For the DFA we will construct, It will keep reading a given Input string 'x' and keeps track of the last two characters it saw.

So our states will represent set of all possible last two character combinations.

⇒ we have 4 states 00, 01, 10, 11 → last 2 chars read in each state.

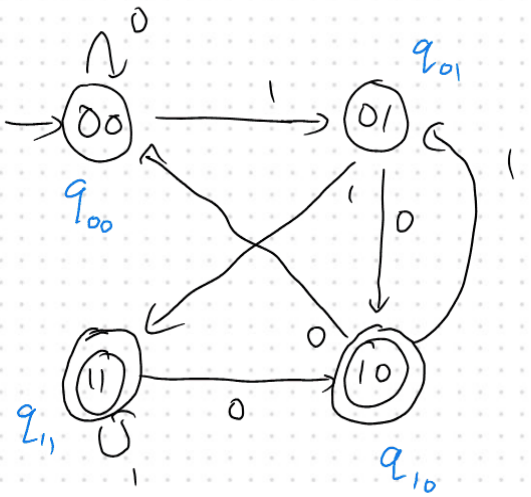
→ Now write the transitions.

$q_{00}, q_{01}, q_{10}, q_{11}$

$\delta(q_{\alpha\beta}, \gamma) = q_{\beta\gamma}$

α, β are both either 0 or 1 each.

γ is 0 (or) 1



→ at end, q_{11}, q_{10} are final states because they end with last 2nd char as 1 and we accept any string that ends there.

→ q_{00} will be our only start state because for string x of length 0 (or) 1 we have no last 2nd char so. we take last 2 chars as 0,0 for reading nothing yet.

$L_8 = \{x \mid x$ is binary representation of all numbers divisible by $3\}$, $\Sigma = \{0,1\}$

We will look at how to keep track of remainder by 3 on binary representations of numbers.

Fact :- if you have binary representation of a number 'n' in a string 'x' made of 0,1's.

→ if you concatenate x with a '0' → x0 will represent number $2n$.

" " " " '1' → x1 " " " $2n+1$.

let that sink in.

and if $n \equiv a \bmod 3$ where $a \in \{0,1,2\}$

then $2n \equiv 2a \bmod 3$ and $2n+1 \equiv (2a+1) \bmod 3$.

For verifying for a given input binary string 'x' is part of our Language means here verifying if the number it is representing has remainder 0 or not. when divided by 3.

→ Now assume we are starting with empty string ε and assume its remainder by 3 is 0. Now we are reading our input string character by character and for concatenating every character one-by-one our 3 remainder keeps changing between 0, 1, 2. → next Page continuation.

Consider example strings → $x =$ "01001" $\pmod 3$

$\epsilon \longrightarrow 0$

"0" (or) $\epsilon 0 \longrightarrow 0 \longrightarrow$ (double of prev value + new digit) $\bmod 3$.

$01 \longrightarrow 1 \longrightarrow 2(0)+1$

$010 \longrightarrow 2 \longrightarrow 2(1)+0$

$0100 \longrightarrow 1 \longrightarrow 2(2)+0 \bmod 3$

$01001 \longrightarrow 0$

→ So for reading character by character we keep rotating between $0,1,2 \bmod 3$.

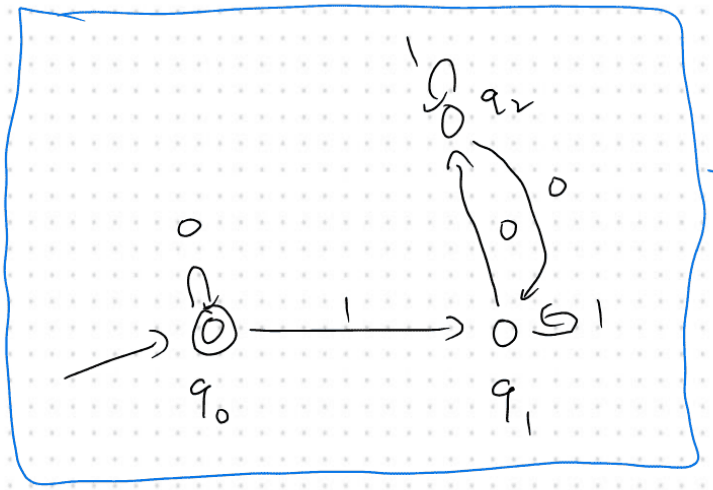
So Construct your DFA with states as q_0, q_1, q_2

where q_i: state where your current $(\bmod 3)$ is i.

and Transition rules are

$$\delta(q_i, a) = q_{(2i+a) \bmod 3}$$

$i \in \{0,1,2\}$, $a \in \{0,1\}$



→ Find DFA which accepts only strings which are binary representations of numbers which are multiples of 3. Including ϵ.

→ q_0 is start state because ϵ has remainder 0 so q_0.

→ q_0 is also final state because we accept only states where remainder is 0.

Similar to L_8 we can construct for divisible by any number.

$L_9 = \{ x \mid x$ is binary rep. of all numbers divisible by $5 \}$, $\Sigma = \{0,1\}$

again your states will be 0 to 5-1 so q_0, q_1, q_2, q_3, q_4

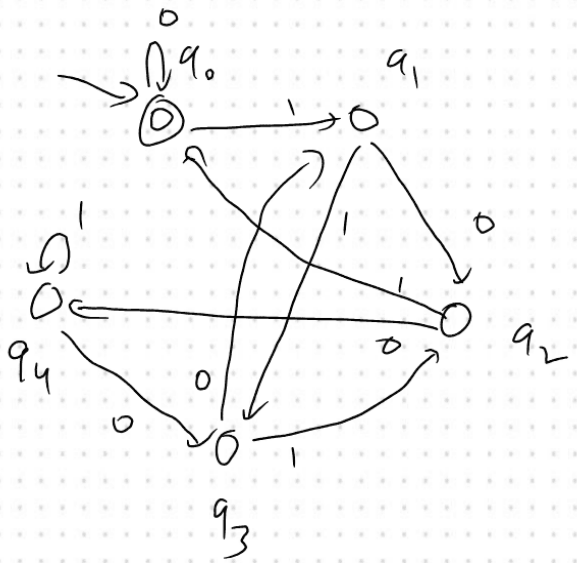
and transitions, same method

$$\delta(q_i, a) = q_{(2i+a) \bmod 5} \quad \text{where } i \in \{0,1,2,3,4\},\ a \in \{0,1\}$$

for making easy, draw transition table based on above δ relation.

	0	1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

from this → draw the diagram



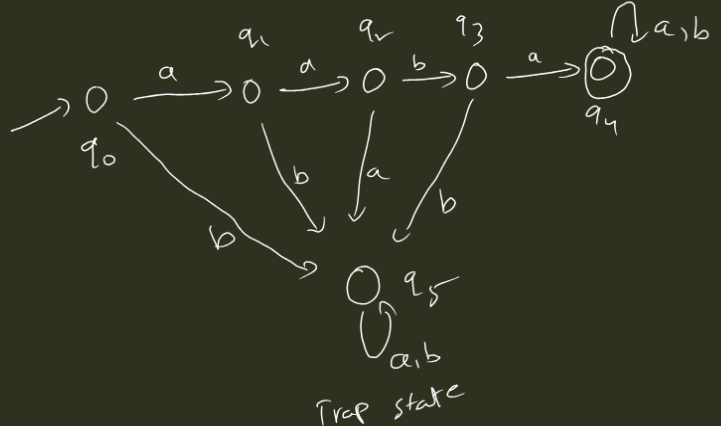
→ start and final states will only be q_0 again, same reason.

→ unless your language is different and says new $L_{q.2}$ accepts only strings which are binary representations of $5n+2$

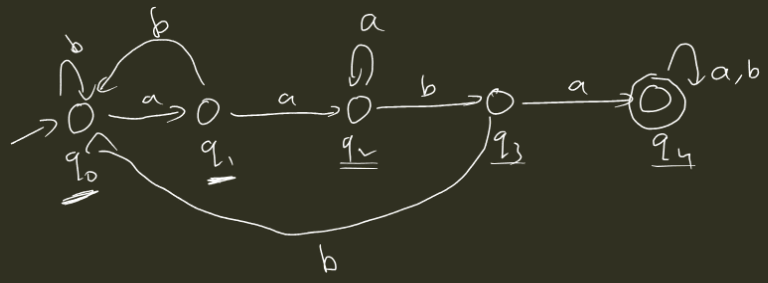
that case you'll make q_2 as final state.

More examples on Constructing DFAs

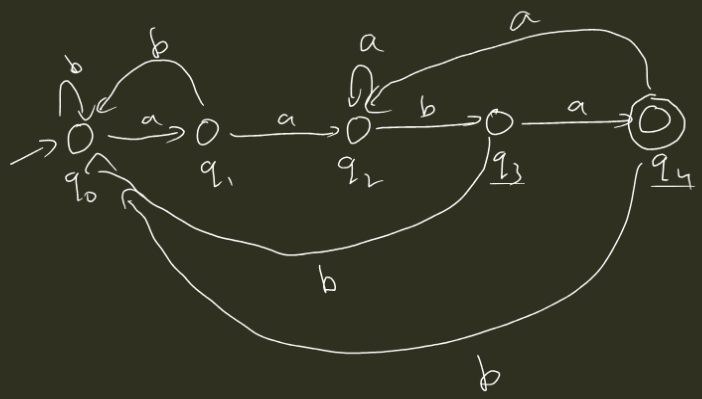
① $L_1 = \{x \mid x \text{ starts with "aaba"}\}$, $\Sigma = \{a, b\}$



② $L_2 = \{x \mid x \text{ contains "aaba" as a substring}\}$, $\Sigma = \{a, b\}$



③ $L_3 = \{x \mid x \text{ ends with "aaba"}\}$, $\Sigma = \{a, b\}$



For every DFA,

1. For every state mention transition for every symbol in Σ.
2. Mention a unique start state.
3. Also mention set of Final states.

x → "aaba" first occurence of "aaba" in x.

q_0 → last characters are 'b' or 'ε'

q_1 → last characters are 'a'

q_2 → " " "aa"

q_3 → " " "aab"

q_4 → " " "aaba"

in L_2
↳ first occurence of "aaba" in x.

in L_3,
↳ x ends with "aaba"

"aabaaba"

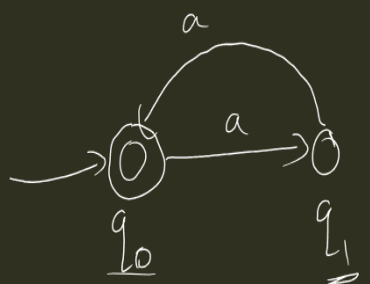
"aaba"ba

"aaba"baaba

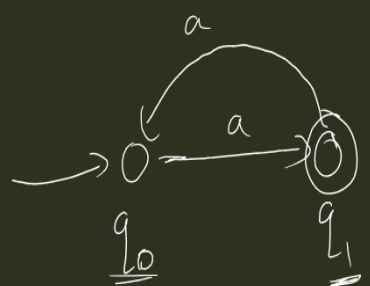
④ $L_4 = \{x \mid x \text{ contains even no. of a's}\}$, $\Sigma = \{a\}$

$\{a^n \mid n = 2k,\ \forall\ k \geq 0,\ k \in \mathbb{W}\}$

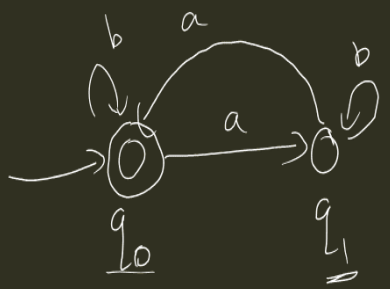
a^n → aaa ... a (n times)



$L_{4,5} = \{a^{2k-1} \mid k \in \mathbb{N}\}$, $\Sigma = \{a\}$



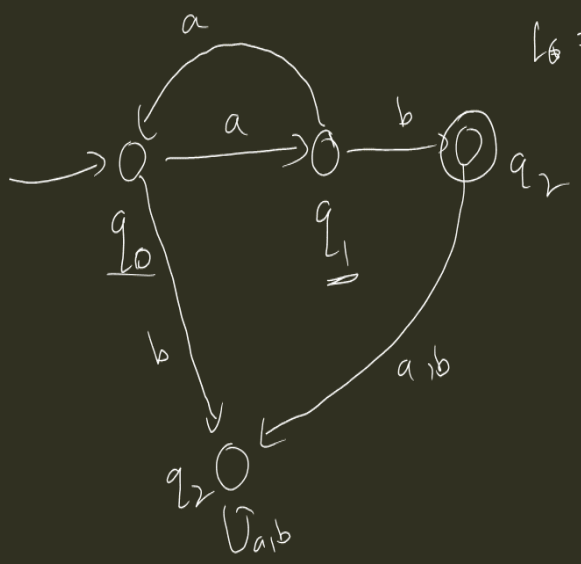
⑤ $L_5 = \{x \mid x \text{ contains even no of a's}\}$, $\Sigma = \{a, b\}$



⑥ $L_6 = \{x \mid x \text{ contains odd no. of a's, and always ends with a 'b'}\}$, $\Sigma = \{a, b\}$

$L_6 = \{a^{2k-1} b \mid k \in \mathbb{N}\}$

$a^{2k-1} b$ → a^{2k} b → a b

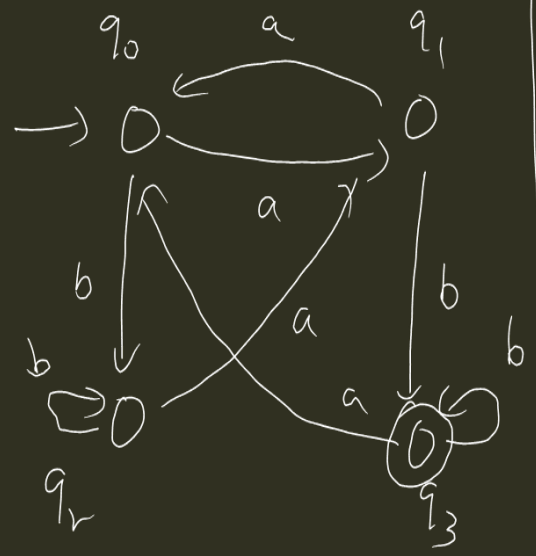


6.1 $L_{6.1} = \{ x \mid x$ contains odd no. of a's and ends with b$\}$, $\Sigma = \{a, b\}$

notice the difference between L_6, $L_{6.1}$

$L_{6.1}$ strings can also contain b's in middle too.

DFA of $L_{6.1}$



Explanation :-

→ q_0 → even no. of a's

→ q_1 → odd no. of a's

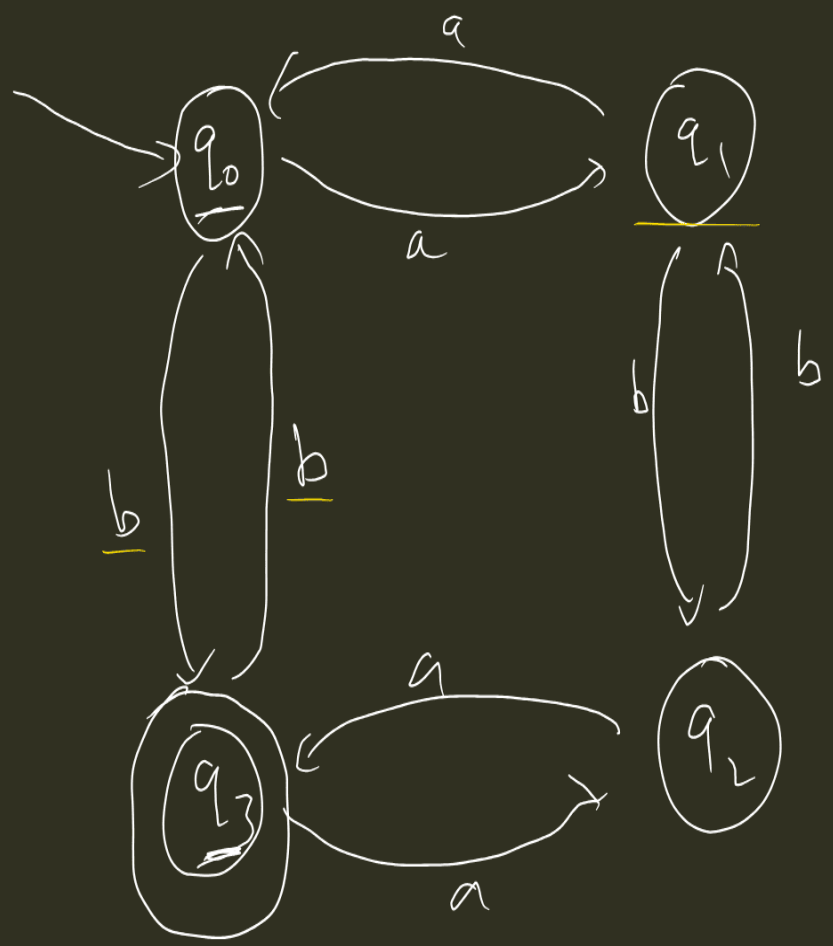
from q_0, q_1 if seen a 'b', move to q_2, q_3 respectively.

→ if kept seeing 'b', stay

→ if seen 'a' go to q_1, q_0 from q_2, q_3 respectively to keep the a's count running.

→ Final state is q_3

⑦ $L_7 = \{x / x$ Contains even no. of a's and odd no. of b's$\}$, $\Sigma = \{a, b\}$



→ if you see even no. of a's

you'll be in q_0 (or) q_3

→ odd no. of a's → q_1 (or) q_2

Stage-1 → q_0, q_1 set

Stage-2 → q_2, q_3 set

→ if odd no. of b's → Stage-2

→ even no. of b's → stage-1

a → odd

b → even

→ q_1 final state

DFA Transition fn δ

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

Path fn ($\hat{\delta}$) $\quad \hat{\delta} : Q \times \Sigma^* \to Q$

$$\hat{\delta}(q, \epsilon) = q$$

$$\hat{\delta}(q, a) = \delta(q, a)$$

$$\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$$

where $w \in \Sigma^*$, $a \in \Sigma$

$$\rightarrow \hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(\epsilon, x), y) \quad \text{where} \quad x, y \in \Sigma^*$$

Now let's look at DFA path function

DFA Path Function

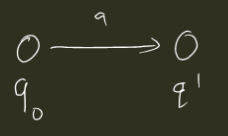
$(Q, \Sigma, q_0, F, \delta)$ → We have to give δ as part of DFA.

↳ Transition fⁿ

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

$\delta(\_, \_) = \_$ ↳ $q' \in Q$

↓ $q \in Q$ ↓ $a \in \Sigma$



Path Function

$\hat{\delta} : Q \times \Sigma^* \to Q$ → $\hat{\delta}$ is defined using δ.

$\hat{\delta}(\_, \_) = \_$ ↳ $q' \in Q$

↓ $q \in Q$ ↓ $x \in \Sigma^*$

↳ if you are at state 'q' upon further reading string 'x' where will you end up (q')

$$\hat{\delta}(q, x) = q'$$

1. $\hat{\delta}(q, \epsilon) = q$
2. $\hat{\delta}(q, a) = \delta(q, a)$ where $a \in \Sigma$
3. $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$ where $w \in \Sigma^*, a \in \Sigma$

↳ all 3 points are simply definition of $\hat{\delta}$.

based on 1 2 3 we can write,

4. $\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$ where $x, y \in \Sigma^*$

Mathematical way of saying $x \in L$ from our DFA Path fⁿ

$$\hat{\delta}(q_0, x) \in F \iff x \in L$$