Unit-2
Introduction to finite Automata
- Ankit Pangeu

DATE

* Finite Automata/ Finite state Machine (FSM):-

Finite Automata (FA) is the simplest machine to recognize patterns. A finite automation is a mathematical (model) abstract machine which has five elements or tuple. It has a set of states and rules for moving from one state to another but it depends upon applied input symbol. Basically it is an abstract model of digital computer.

A finite automation/automata consists of five tuples $(Q, \Sigma, \delta, q_0, F)$ where,

$Q \rightarrow$ Finite set of states

$\Sigma \rightarrow$ Alphabet (set of input symbols)

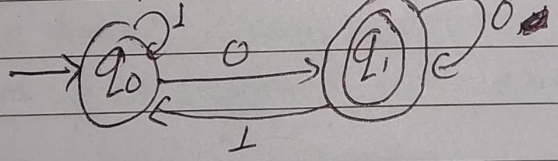
$\delta \rightarrow$ Transition function ~~$\delta: Q \times \Sigma \rightarrow Q$~~

$q_0 \rightarrow$ Initial state.

$F \rightarrow$ Set of final states: $F \subseteq Q$

(F is a subset of Q)

Eg: We have following FSM



$Q = \{q_0, q_1\}$ = set of all states

$\Sigma = \{0, 1\}$ = alphabet (set of all input)

$\delta = Q \times \Sigma$

q_0 = initial state.

$F = \{q_1\}$ Note: Initial state is only one but final states can be more than one

$\delta = Q \times \Sigma$

	0	1
$\rightarrow q_0$	q_1	q_0
$* q_1$	q_1	q_0

δ: transition table.

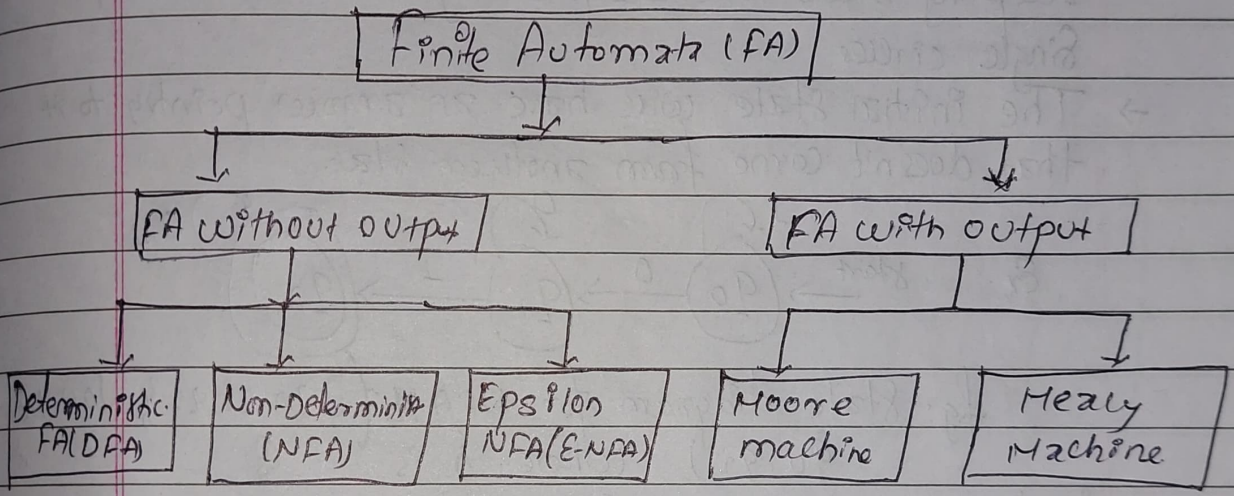
PAGE

DATE

* Application of FSM:

The finite state machines are used in applications in Computer science and data networking. Ex: finite-state machines are basis for programs for spell checking, indexing, grammar, searching large bodies of text, recognizing speech, etc.

* Categories of FSM



1) Deterministic Finite Automata (DFA)

A DFA is defined as five tuple $(Q, \Sigma, \delta, q_0, F)$ where,

Q: Finite set of states, Σ, δ, F:

δ: transition function:

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

→ It is the simplest model of computation which has a very limited memory.

→ In DFA, for a particular input character, the machine goes to one state only. A transition function is defined on every state for every input symbol.

DATE

* Notations for DFA (Methods of describing transition functions)

There are two preferred notations for describing this class of automata.

1. Transition Diagram (State diagram)

It is a graphical representation in which states are represented by circles, transitions are represented by arrows with input symbols as labels and accepting states are designated by double instead of single circles.

→ The initial state will have an arrow pointing to it that doesn't come from another state.

Ex:

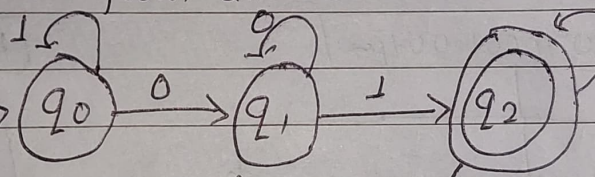


fig: State diagram for DFA $M = \{Q, \Sigma, \delta, q_0, F\}$ where $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0, 1\}$, δ, q_0, F

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

δ:	0	1
→ q_0	q_1	q_0
q_1	q_1	q_2
* q_2	q_2	q_2

fig: Transition table. (not necessary here)

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

DATE

2) Transition Table:

Transition table is a tabular representation of the transition function δ that takes the arguments from $Q \times \Sigma$ and returns one of the states in Q.

→ The row of the table corresponds to the states

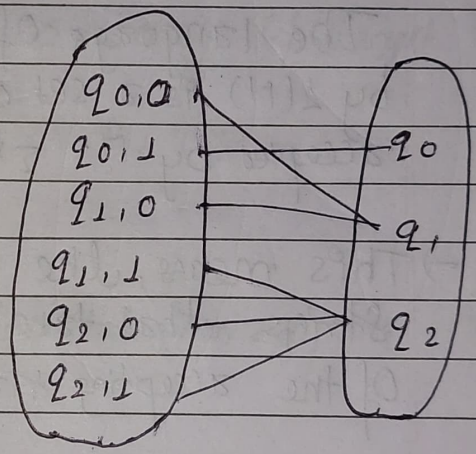
→ The column of the table corresponds to the ~~state~~ input symbol.

→ The starting state is indicated by → and final or accept state is indicated by *.

Ex: For above example, transition table is

δ:	0	1
→ q_0	q_1	q_0
q_1	q_1	q_2
* q_2	q_2	q_2

or =



As a whole example.

Ex: A D.finite automation that accepts strings over $\Sigma = \{0,1\}$ that are length at least 2.

=) Such a DFA is a 5-tuple:

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

$$L = \{00, 01, 10, 11, 000, 001, \ldots\}$$

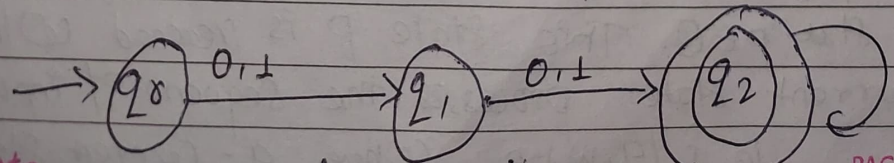


fig: State (transition) diagram

classmate

PAGE

DATE

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

	δ:	0	1
→	q_0	q_1	q_1
	q_1	q_2	q_2
*	q_2	q_2	q_2

fig: State transition table

* Language of DFA

The language of DFA, $M = (Q, \Sigma, \delta, q_0, F)$ denoted by $L(M)$ is a set of strings over Σ^* that are accepted by M. Σ^* is a set of all input symbols.

→ This means, the language of DFA is the set of all strings that take DFA from start state to one of the accepting state

* Extended Transition Function of DFA ($\hat{\delta}$):

The extended transition function is a transition function that tells the DFA, which state to go after reading a string from the particular state.

→ Formally, it is the transition function of DFA, that takes two arguments as input, one is state q of Q & another is a string $w \in \Sigma^*$ & generates a state $p \in Q$. This state p is reached when the current state processes the sequence of inputs.

ie $\hat{\delta}(q, w) = p$ where, q = Current state, w = Sequence of inputs, p = generated or new received state

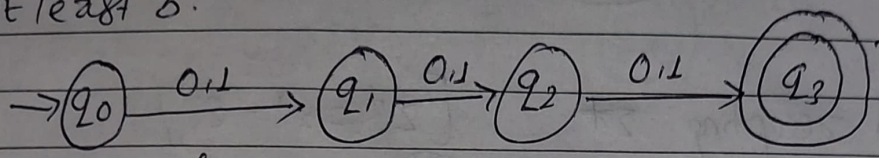
classmate PAGE

DATE

Some examples of DFA

Q. Construct a DFA accepting strings over $\Sigma = \{0, 1\}$ of length at least 3.

⇒



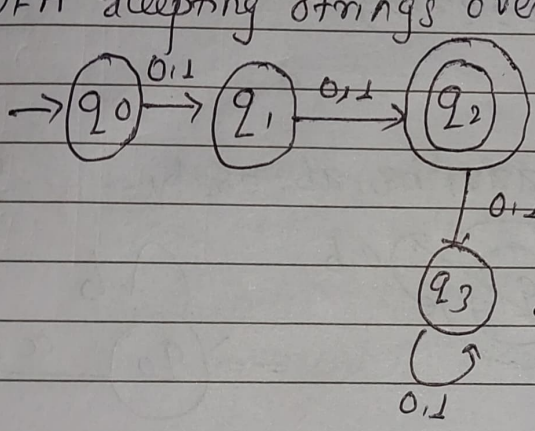
$L = \{000, 001, 100, 101, 0100, 1001, 10000, \ldots\}$

$\therefore M = \{Q, \Sigma, \delta, q_0, F\}$

$= (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, q_3)$

δ defined by above transition diagram

Q. DFA accepting strings over $\Sigma = \{0, 1\}$ of length exactly 2

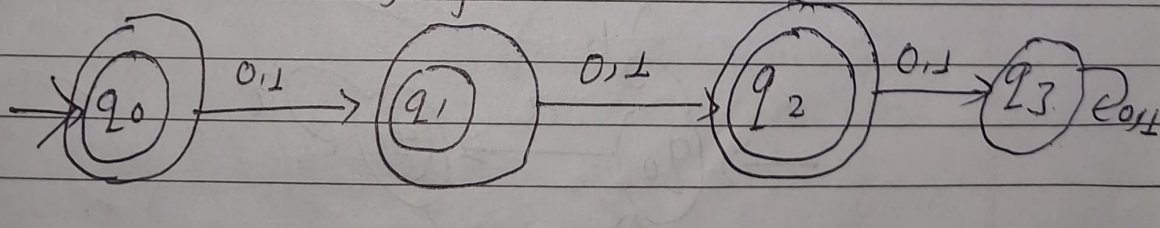


Q. DFA accepting strings over $\Sigma = \{0, 1\}$ of length at most 2.

⇒ $L = \{\epsilon, 0, 1, 00, 01, 10, 11\}$

It is a finite language.

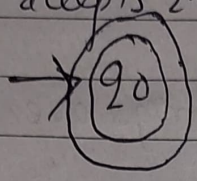
Here. L = language of DFA



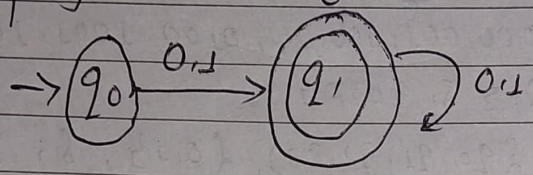
DATE

Q. DFA accepting all strings over $\Sigma = \{a, b\}$.

⇒ ie. DFA accepts Σ^*



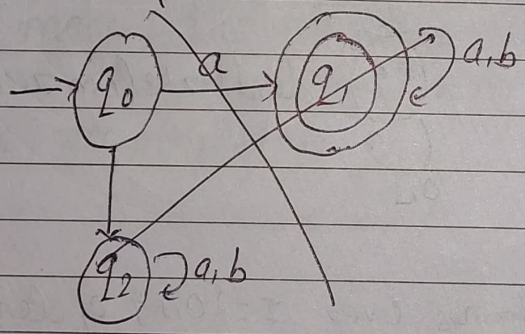
Q. DFA accepting Σ^+ $[\Sigma^+ = \Sigma^* - \varepsilon]$

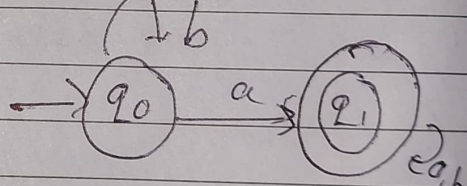


Q. Construct a DFA which accepts a language of all strings over $\Sigma = \{a, b\}$

→ containing 'a'

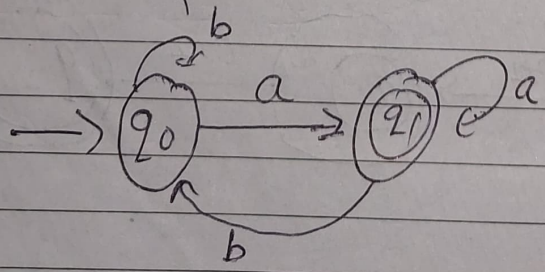
$L = \{a, aa, aaa, ba, ab, abab, \ldots\}$





→ end with 'a'

$L = \{a, aa, aaa, ba, bba, baba, aba, \ldots\}$



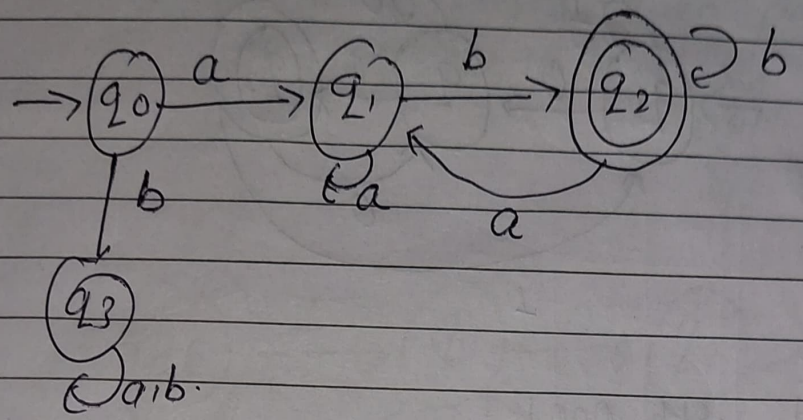
classmate

PAGE

DATE

-) Starting with 'a' and ending with 'b'

$L = \{ab, abab, ababab, aaab, abbbb, \ldots\}$

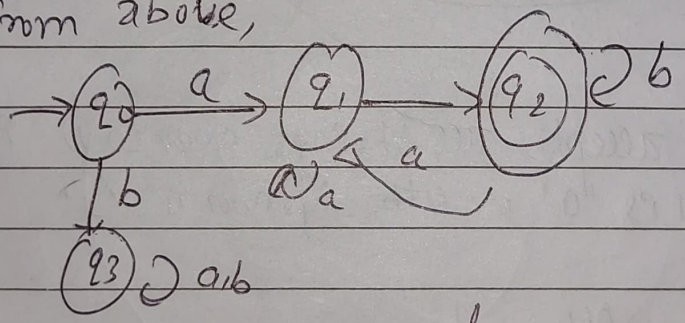


→ not starting with 'a' or not ending with 'b'

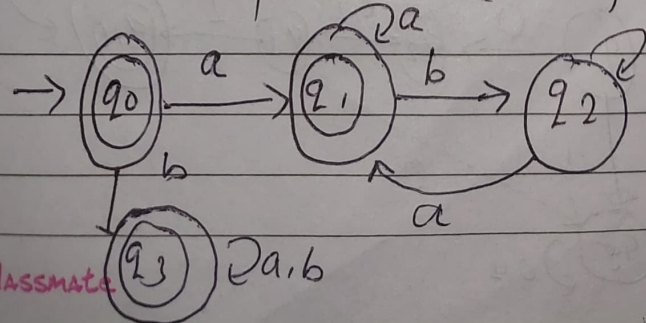
$L = \{\epsilon, a, b, \ldots\}$

If A = not starting with 'a'
B = not starting with 'b'
then, $(A \cup B)^c = A^c \cap B^c$
so, its complement is starting with a & ending with b

From above,



Now, its complement is: (final state = non final, non final = final state)



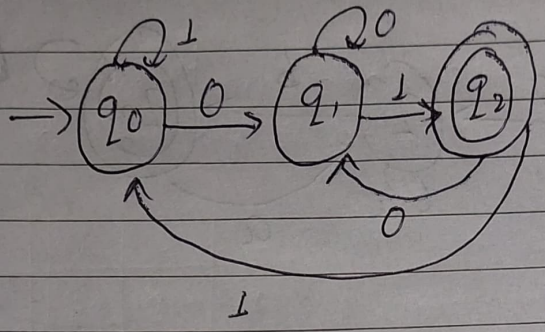
classmate

PAGE

DATE

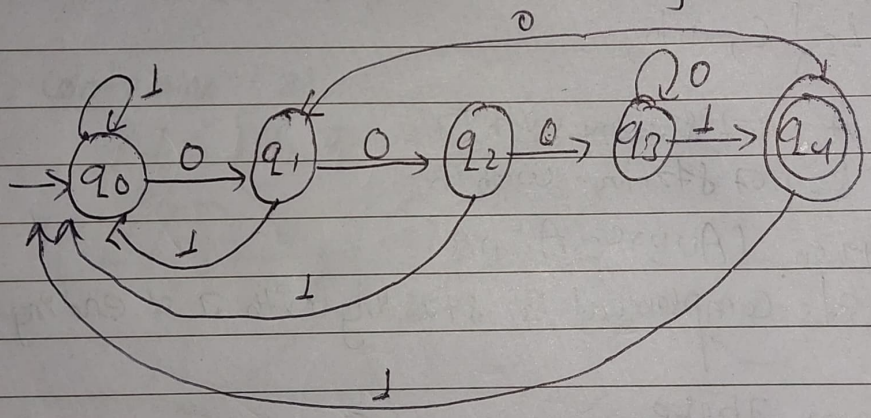
Q. DFA accepting string over $\Sigma = \{0,1\}$ that end with 001

⇒ $L = \{01, 001, 1101, 1001, 0001, 1101, 110001, \ldots\}$



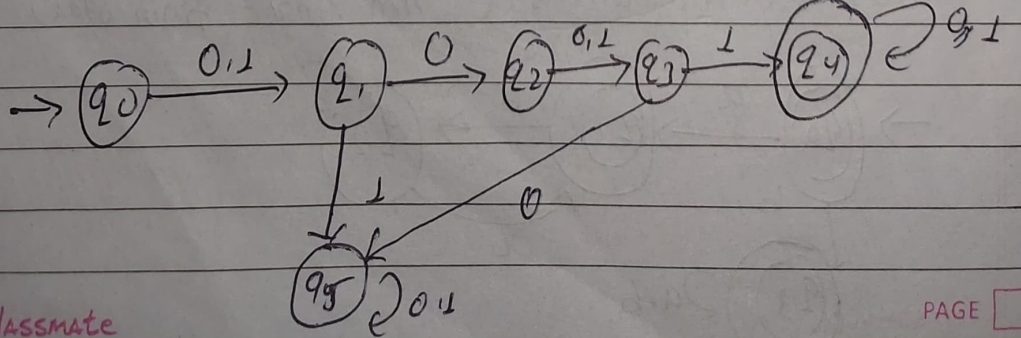
→ end with 0001

$L = \{0001, 10001, 00001, 110001, 0011 0001, 0101 0001, \ldots\}$



Q. DFA which accepts all strings over (0,1) in which 2nd symbol is '0' & 4th symbol is '1'

01 0 01 1



classmate

PAGE

DATE

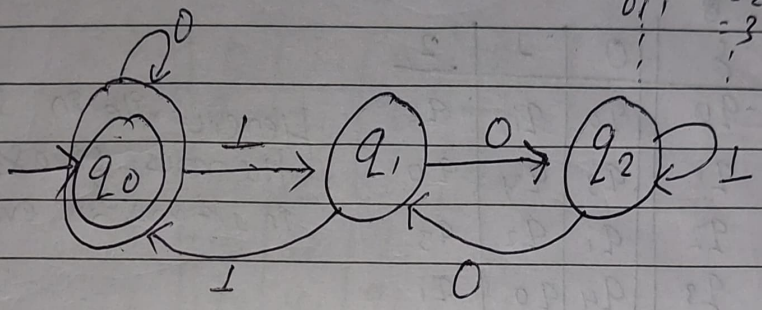
Q Design a DFA which accept a language of all binary strings divisible by three over $\Sigma(0,1)$ [NOTE: base = 2]

=) When any no. is divided by 3, then possible remainder

ex: $L = \{0, 00, 000, 011, \ldots\}$

q_0 0
q_1 1
q_2 2

000 = 0
001 = 1
010 = 2
011 = 3
⋮ ⋮



δ	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

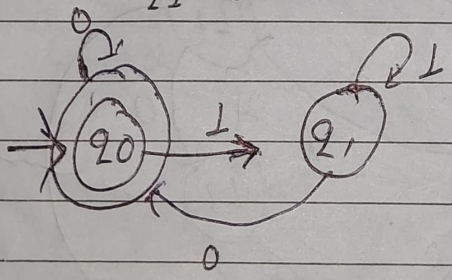
Q. n: is a set of natural numbers. Design a dfa that accept a string

i) Divisible by 2
ii) Divisible by 2 and 3

i) Divisible by 2

q_0 0
q_1 1

$L = \{0, 00, 000, 10, 100, \ldots\}$



ii) Divisible by 2 & 3

DATE

Q. Design a DFA that accepts the strings divisible by 5 over $\Sigma = \{0,1,2\}$ [Note: base = 3]

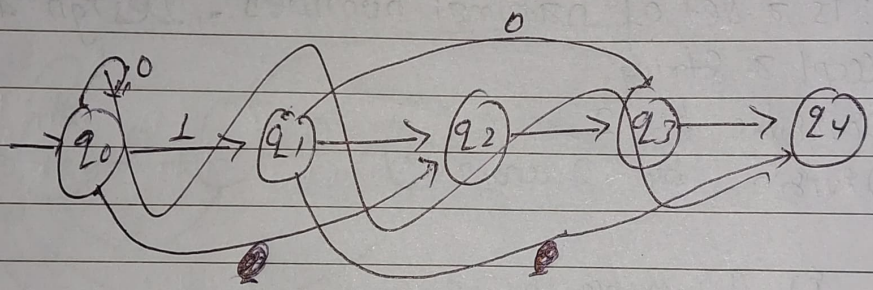
=) Trick:

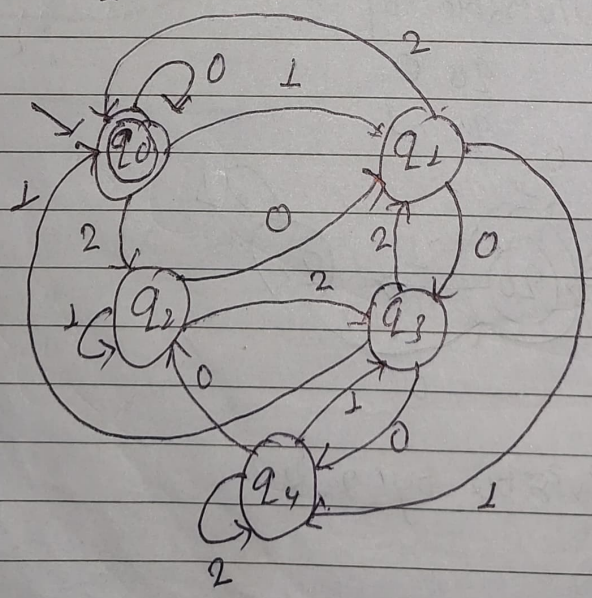
1st make transition table:

It is divisible by 5, so it has minimum of 5 states

δ	0	1	2
q_0	q_0	q_1	q_2
q_1	q_3	q_4	q_0
q_2	q_1	q_2	q_3
q_3	q_4	q_0	q_1
q_4	q_2	q_3	q_4

Here, write in sequence in the rows, like start from q_0 in 1st row, fill every row.



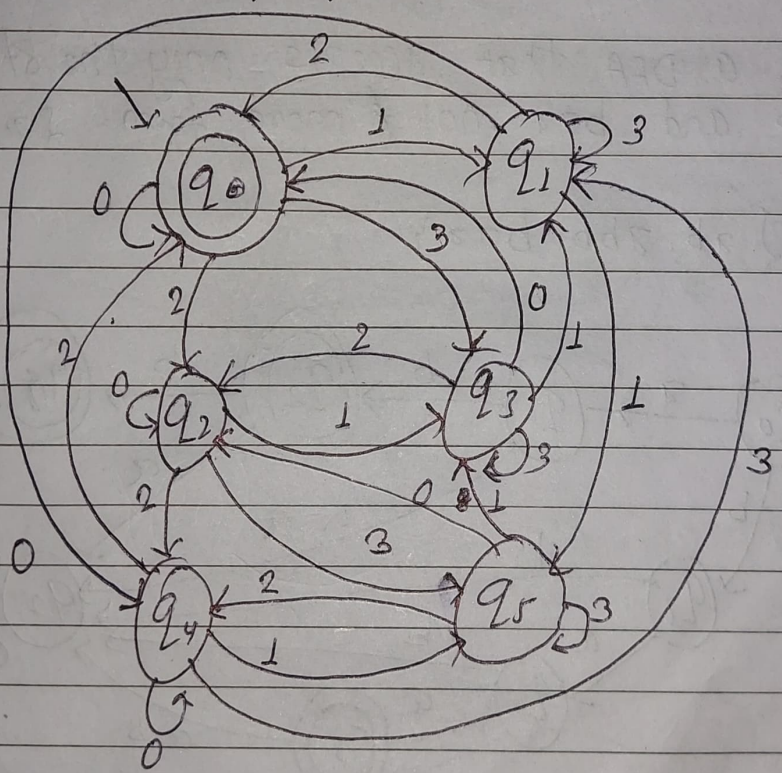


DATE

Q. Design a DFA that accepts the strings divisible by 6 over $\Sigma = \{0,1,2,3\}$. [Note: base = 4]

⇒ Here, divisible by 6, So it has at least 6 states. Let them be $q_0, q_1, q_2, q_3, q_4, q_5$,

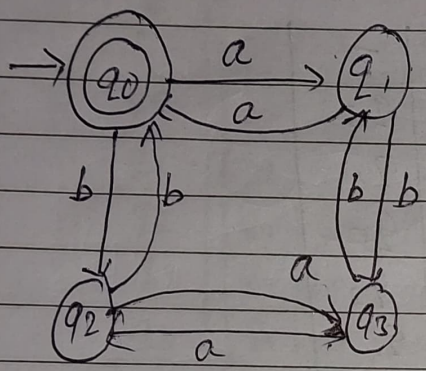
δ	0	1	2	3
q_0	q_0	q_1	q_2	q_3
q_1	q_4	q_5	q_0	q_1
q_2	q_2	q_3	q_4	q_5
q_3	q_0	q_1	q_2	q_3
q_4	q_4	q_5	q_0	q_1
q_5	q_2	q_3	q_4	q_5



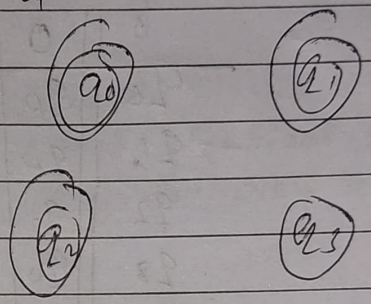
DATE

Q. Construct a DFA that accepts all the strings of alphabet {a,b} having each strings with even number of a's and even number of b's. (2076, 2075)

=> L = { ϵ, aa, bb, abab, aabb, bbaa, baba, abba, baab, ...

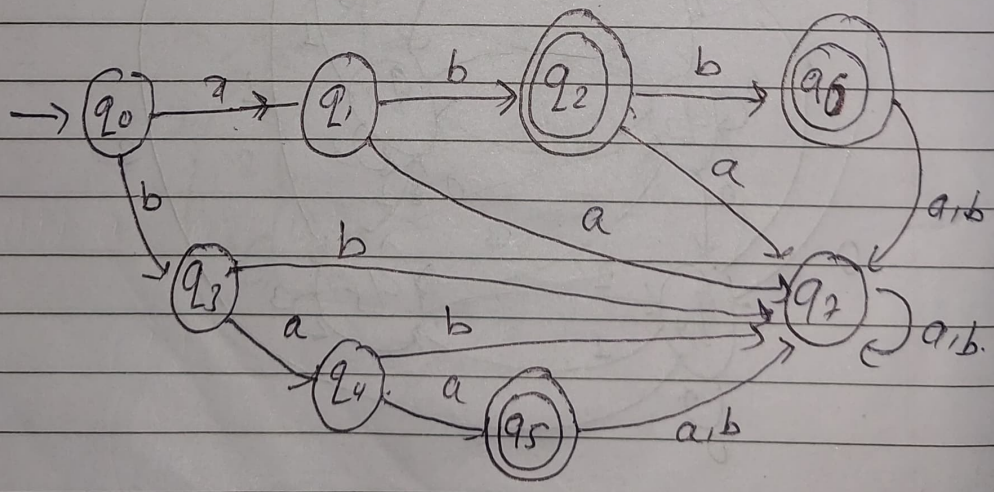


If a's even or b's even



Q. Construct a DFA that accepts only the strings ab, abb and baa not more from {a,b}

=> L = { ab, abb, baa }

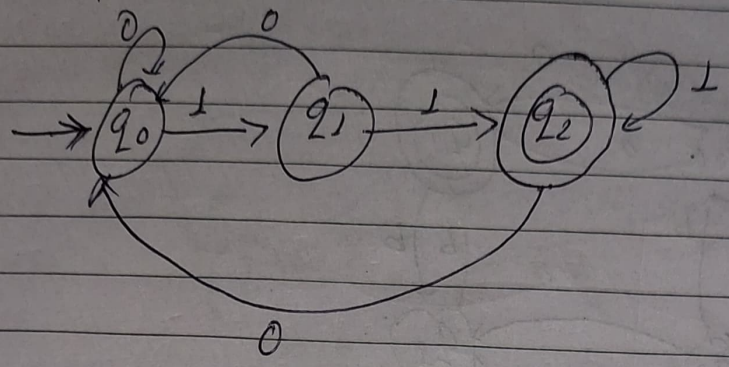


Q.

DATE

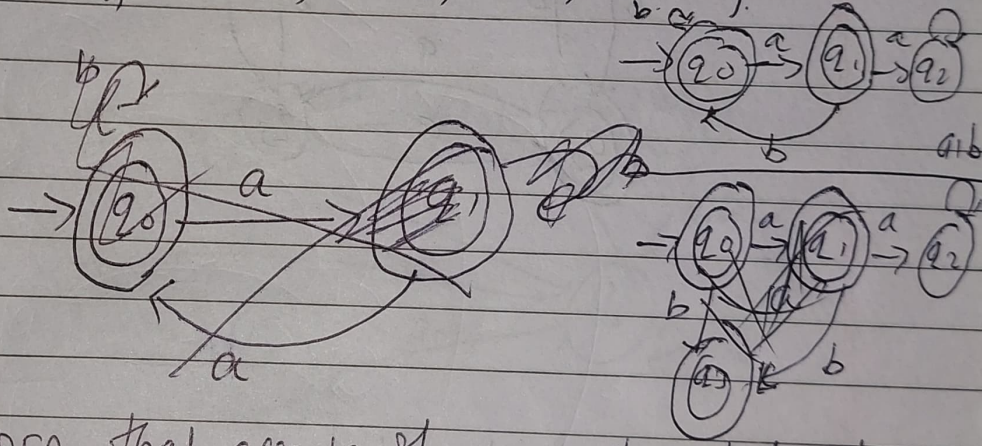
Q. Give a DFA for language of strings over {0,1} where each strings ends with 11

⇒ $L = \{11, 011, 111, 0011, 1011, \dots\}$



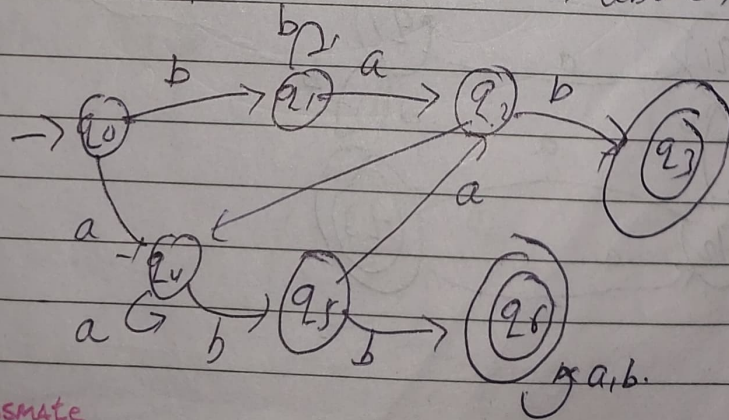
Q Give the DFA for language of strings over {a,b} where no two consecutive a's occured.

⇒ $L = \{\epsilon, a, b, ab, abb, abab, aba, \dots\}$



Q. Design a DFA that accepts strings with substring bab or abb. $L = \{bab, abb, abab, abbab, bbabb, \dots\}$

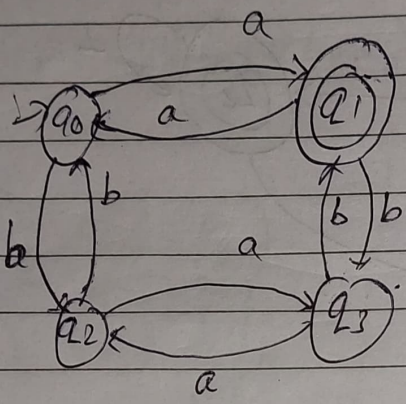
⇒



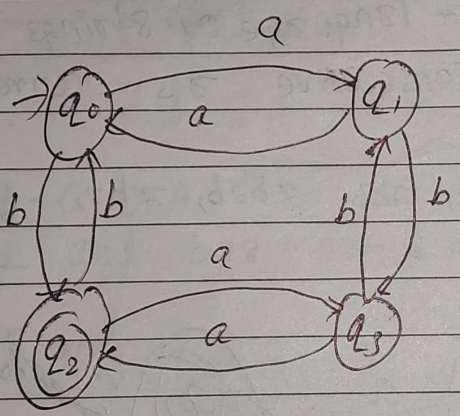
DATE

Q. Construct a dfa over $\Sigma = \{a, b\}$ that accepts strings with

- odd a and even b

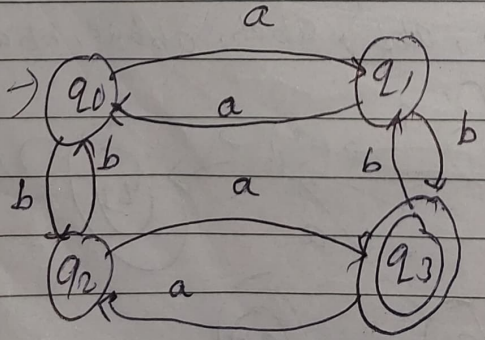


- Even a and odd b.



bab

- Odd a and odd b.



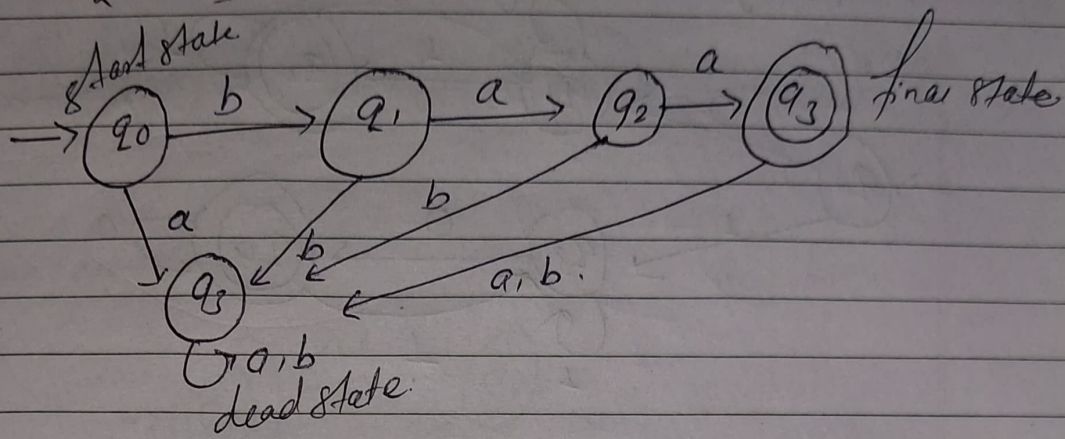
classmate

PAGE

DATE

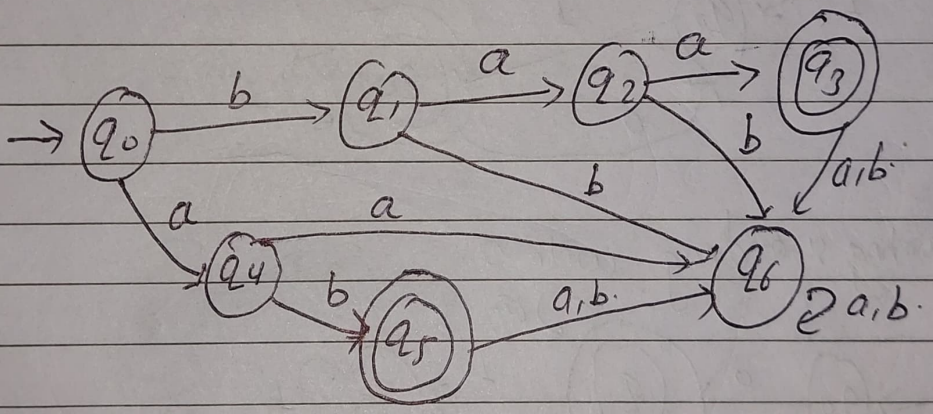
Q. Construct a DFA that accepts only baa over $\Sigma = \{a, b\}$

⇒ $L = \{baa\}$



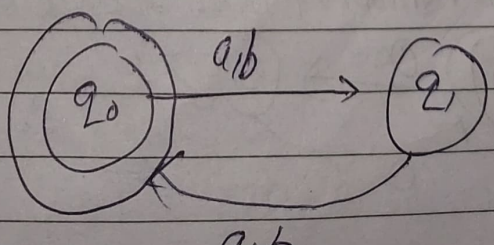
Q. Construct a DFA that accepts only baa and ab. over $\Sigma = \{a, b\}$

⇒ $L = \{baa, ab\}$



Q. Construct a DFA that accepts strings of length divisible by 2

$L = \{ \epsilon, aa, ba, bb, ab, abba, \dots \}$

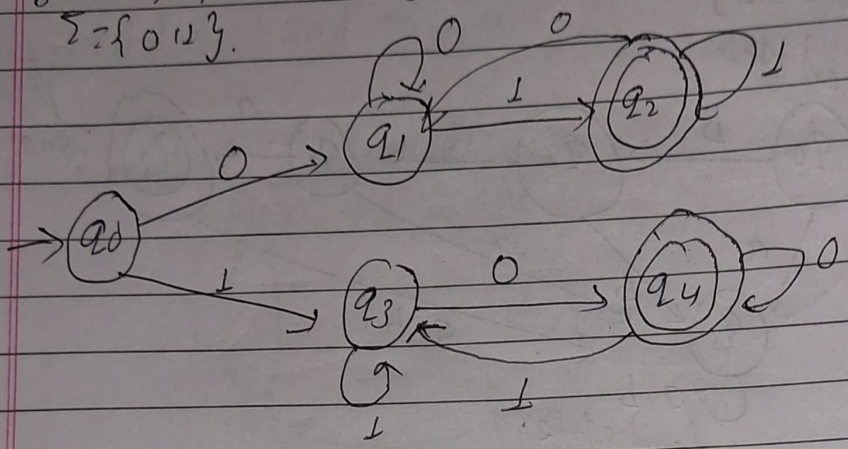


classmate

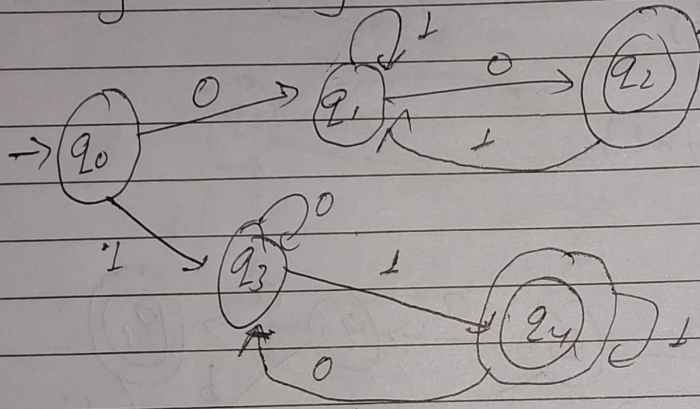
PAGE

DATE

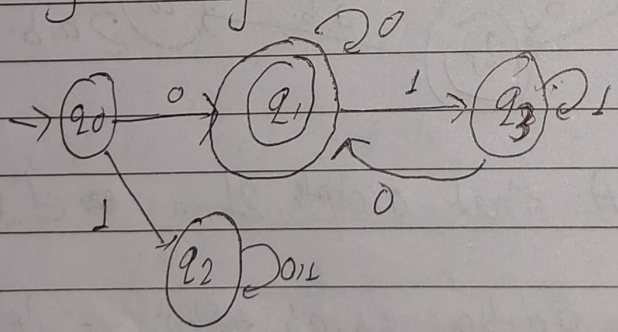
Q. Draw a DFA for language accepting strings starting & ending from different character over $\Sigma = \{0, 1\}$.



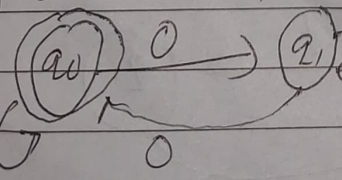
Q. ending & starting with same character



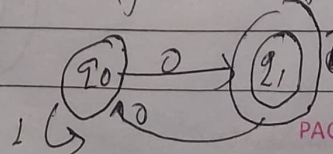
Q. starting & ending with 0



Q. accepting even number of zeros strings over $\Sigma = \{0, 1\}$



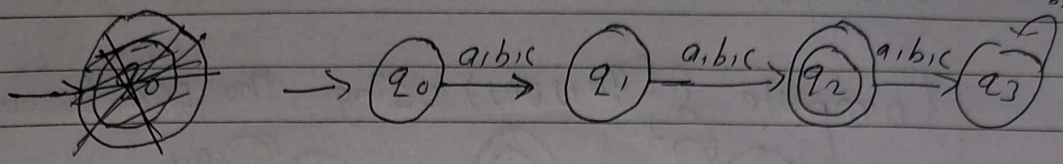
if odd no. of zeroes.



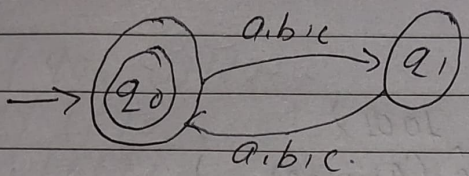
DATE

Q. Given $\Sigma = \{a, b, c\}$, construct a DFA that accepts strings over Σ of length 2

=>

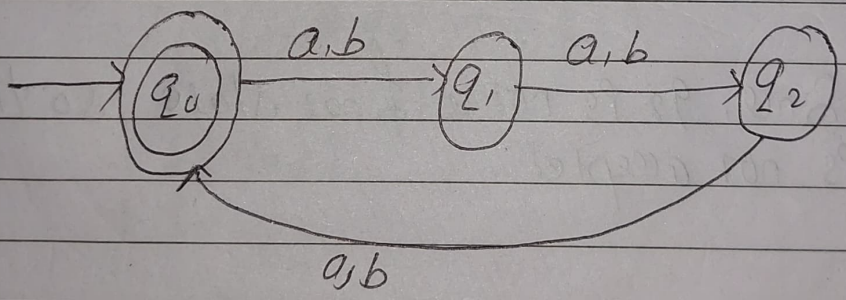


even length strings



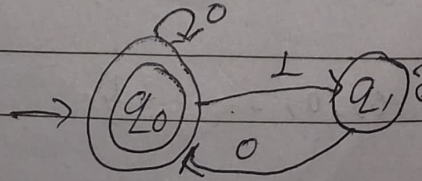
Q. Given $\Sigma = \{a, b\}$. construct a DFA that accepts strings over Σ such that their length are divisible by 3.

=> $L = \{\epsilon, aaa, baa, bbb, aabbb, ----\}$



Q. Given $\Sigma = \{0, 1\}$. construct a DFA that accepts all strings over Σ s.t. ~~those string~~ its equivalent binary num. is divisible by (2).

=> $L = \{\epsilon, 0, 00, 000, 100, 00100, ----\}$



classmate

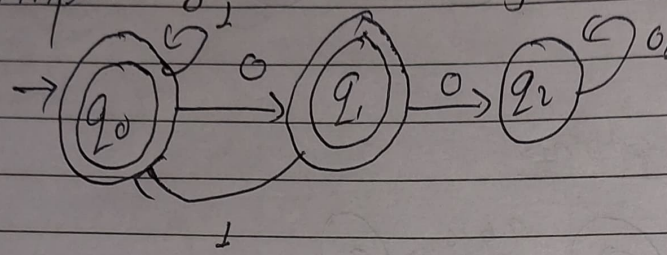
PAGE

DATE

* Extended Transition function of DFA ($\hat{\delta}$):-

Example:

Compute $\hat{\delta}(q_0, 1001)$ using the DFA below



Here, $\hat{\delta}(q_0, 1001)$

$= \delta(\hat{\delta}(q_0, 100), 1)$

$= \delta(\delta(\hat{\delta}(q_0, 10), 0), 1)$

$= \delta(\delta(\delta(\hat{\delta}(q_0, 1), 0), 0), 1)$

$= \delta(\delta(\delta(\delta(\hat{\delta}(q_0, \epsilon), 1), 0), 0), 1)$

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

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

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

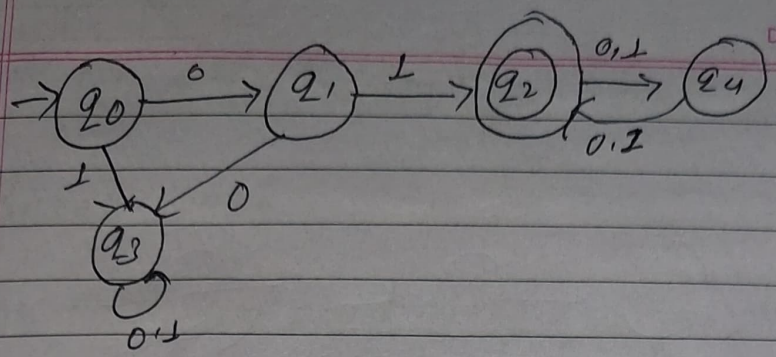
$= \delta(q_2, 1)$

$= q_2$

Since, q_2 is not a final state, so the string 1001 is not accepted.

* Given $\Sigma = \{0, 1\}$. Construct a DFA accepting all strings over Σ such that the strings begin with (01) and are of even length.

→ $L = \{01, 0111, 0110, \ldots\}$



Let us stimulate $\hat{\delta}$ on string "01101"

$\hat{\delta}(q_0, 01101)$

$\Rightarrow \delta(\hat{\delta}(q_0, 0110), 1)$

$\Rightarrow \delta(\delta(\hat{\delta}(q_0, 011), 0), 1)$

$\Rightarrow \delta(\delta(\delta(\hat{\delta}(q_0, 01), 1), 1), 0), 1)$

$\Rightarrow \delta(\delta(\delta(\delta(\hat{\delta}(q_0, 0), 1), 1), 1), 0), 1)$

~~$\Rightarrow \delta(\delta(\delta(\delta(\delta(q_0, \epsilon), 0), 1), 1), 1), 0), 1)$~~

$\Rightarrow \delta(\delta(\delta(\delta(\delta(\delta(q_0, \epsilon), 0), 1), 1), 1), 0), 1)$

$\Rightarrow \delta(\delta(\delta(\delta(\delta(q_0, 0), 1), 1), 1), 0), 1)$

$\Rightarrow \delta(\delta(\delta(\delta(q_1, 1), 1), 1), 0), 1)$

$\Rightarrow \delta(\delta(\delta(q_2, 1), 1), 0), 1)$

$\Rightarrow \delta(\delta(q_4, 1), 0), 1)$

$\Rightarrow \delta(q_2, 0), 1)$

$\Rightarrow \delta(q_4, 1)$

$\Rightarrow q_2$

Since, we land on q_2, the accepting state after reading 01101 is q_2, so the string 01101 is accepted by DFA

DATE

Non-Deterministic finite Automata (NFA)

Background:

In DFA, we could make a unique transition from a particular state on reading an input symbol. So, acceptance of a string by a DFA produces a unique transition path.

In a NFA, multiple transitions on reading some input symbol at some particular state are allowed. ie. A NFA can be at multiple states at a time. Similarly, at a particular state no transition might be defined for input symbols. Such a situation is called dead configuration or dead transition.

Formal Definition:

A NFA is defined as a mathematical model that consists of 5 tuples (quintuple) $(Q, \Sigma, \delta, q_0, F)$

where

Q = set of finite states

Σ = set of input symbol (finite)

q_0 = Initial or start state $q_0 \in Q$.

F = set of final states, $F \subseteq Q$

δ = $Q \times \Sigma \rightarrow 2^{|Q|}$

which is a transition function

There are two main things that we should keep in mind.

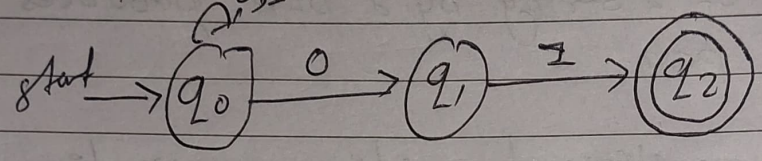
PAGE

DATE

(i) NFA does not contain any dead state.

(ii) Unlike DFA, a transition function in NFA takes the NFA from one state to several states just with a single input.

Example: A NFA that accepts strings that end with 01 over $\Sigma = \{0,1\}$

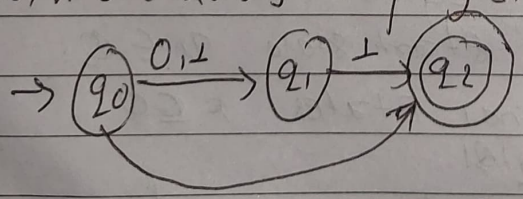


Here, q_0 state on getting 0 goes to q_0 itself as well as q_1. But in DFA, on getting particular input it goes to one particular state only

* Notations for NFA & Notations for NFA are also same as DFA. Write same theory but give examples of NFA

* Language of NFA: Same as DFA, except for definition of δ, we use $\delta: Q \times \Sigma \to 2^{|Q|}$.

Ex: NFA over $\{0,1\}$ accepting strings $\{01, 011, 1\}$

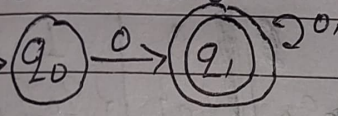
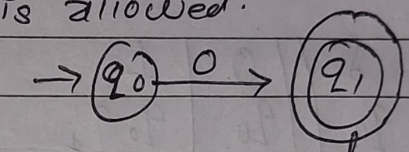
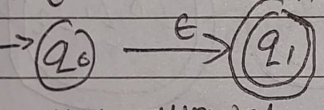
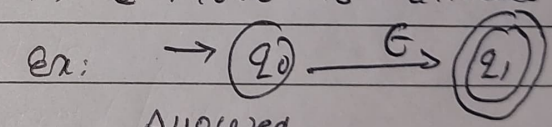


transition table

δ	0	1
→ q_0	$\{q_0, q_1\}$	q_1
q_1	$\emptyset$	q_2
* q_2	$\emptyset$	$\emptyset$

Difference between DFA & NFA

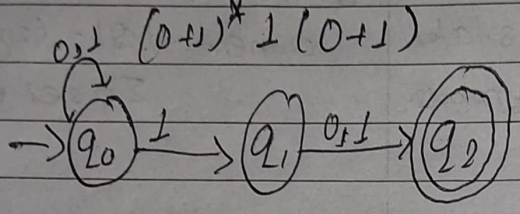
DATE

	DFA	NFA
i)	It is a mathematical model consisting of 5 tuples, where transition function δ is defined as: $\delta: Q \times \Sigma \to Q$ where, Q = set of finite states, Σ = set of input symbols.	i) It is a mathematical model consisting of 5 tuples where transition function δ is $\delta: Q \times \Sigma \to 2^{\lvert Q \rvert}$ where, $\lvert Q \rvert$ = length of Q, Σ = set of input symbols
ii)	In DFA, dead transition is not allowed. Ex:  Here, transition for input 1 in q_0 is not defined, which is not allowed.	ii) In DFA, dead transition is allowed.  Here, transition for input 1 in q_0, & input 0,1 is q_1 is not defined, which is allowed in NFA
iii)	Multiple choices are not available corresponding to an input.	ii) Multiple transition choices are available corresponding to an input.
iv)	(Null) ϵ-Move is not allowed. Ex:  Not allowed.	i) ϵ-Move is allowed Ex:  Allowed
v)	Digital computers are deterministic.	v) Non deterministic feature is not associated with real computers. They are just theoretical concept.
vi)	Designing and understanding is difficult. (not for all cases but some cases) & no. of states might be more.	vi) Designing & understanding is easy. & no. of states will be less

DATE

Q. Design a NFA of all binary strings in which 2nd last bit is 1 over $\Sigma = \{0, 1\}$

⇒ $L = \{10, 11, 010, 110, 0110, \ldots\}$

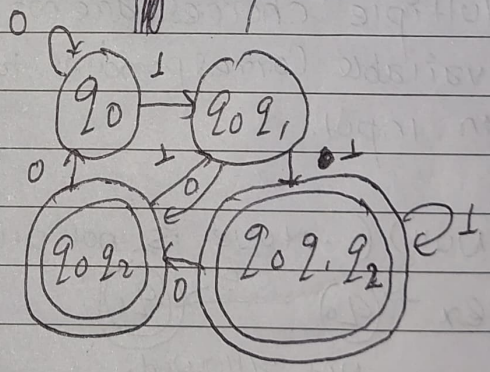


To convert this NFA to DFA, 1st make transition table

	δ	0	1
→	q_0	q_0	$q_0 q_1$
	q_1	q_2	q_2
*	q_2	-	-

⇒

		0	1
→	q_0	$\{q_0\}$	$\{q_0, q_1\}$
	$q_0 q_1$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
*	$q_0 q_2$	$\{q_0\}$	$\{q_0, q_1\}$
*	$q_0 q_1 q_2$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$



DFA

NOTE: Table में No comma

DATE

Extended transition function of NFA ($\hat{\delta}$)

It is defined as follows.

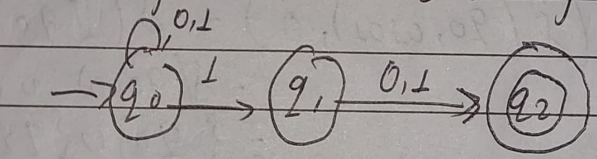
BASIS: $\hat{\delta}(q, \epsilon) = q$

INDUCTION:

- Suppose w is a string of the form $w = xa$, where a is the final symbol of w and x is the rest of w.
- Also suppose $\hat{\delta}(q, x) = \{p_1, p_2, \ldots, p_k\}$
- Let $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$

Then $\hat{\delta}(q, w) = \{r_1, r_2, \ldots, r_m\}$

Ex: Given, $\Sigma = \{0, 1\}$, construct a NFA accepting set of all strings whose 2nd last symbol is 1.



* This NFA accepts the string "01010".
* How does "01010" get accepted?

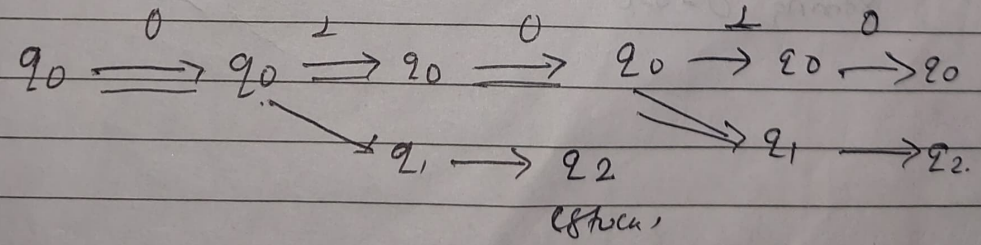


Fig: Transition tree

DATE

use extended transition function for "01010"

$\hat{\delta}(q_0, \epsilon) = \{q_0\}$

$\hat{\delta}(q_0, 0) = \delta(\hat{\delta}(q_0, \epsilon), 0) = \delta(\{q_0\}, 0) = \delta(q_0, 0) = \{q_0\}$

$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1) = \delta(\{q_0\}, 1) = \{q_0, q_1\}$

$\hat{\delta}(q_0, 010) = \delta(\hat{\delta}(q_0, 01), 0) = \delta(\{q_0, q_1\}, 0)$

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

$= q_0 \cup q_2$

$= \{q_0, q_2\}$

$\hat{\delta}(q_0, 0101) = \delta[\hat{\delta}(q_0, 010), 1) = \delta(\{q_0, q_2\}, 1)$

$= \delta(q_0, 1) \cup \delta(q_2, 1)$

$= \{q_0, q_1\} \cup \{\phi\}$

$= \{q_0, q_1\}$

$\hat{\delta}(q_0, 01010) = \delta(\hat{\delta}(q_0, 0101), 0) = \delta(\{q_0, q_1\}, 0)$

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

$= \{q_0\} \cup \{q_2\}$

$= \{q_0, q_2\}$

Since, q_2 is an accept state, we accept the string 01010

DATE

* Language of NFA (based on extended trans. func)

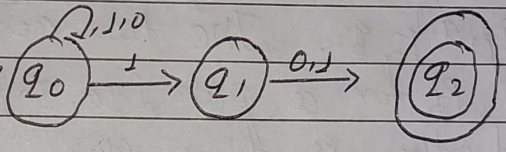
Given a NFA $N = \{Q, \Sigma, \delta, q_0, F\}$, the language accepted by it is defined as follows:

$$L(N) = \{ w \mid \hat{\delta}(q_0, w) \cap F \neq \phi \}$$

It means, the language of N consists of such strings w such that the set of ~~strings~~ states the NFA will be in after reading w contains at least one final state.

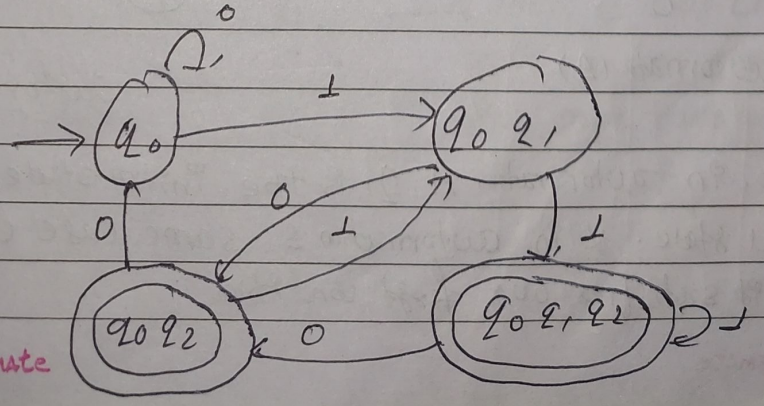
* Conversion of NFA to DFA using {subset construction method}

Ex:



	0	1
→ $[q_0]$	$[q_0]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_2]$	$[q_0, q_1, q_2]$
* $[q_0, q_2]$	$[q_0]$	$[q_0, q_1]$
* $[q_0, q_1, q_2]$	$[q_0, q_2]$	$[q_0, q_1, q_2]$

Note: Max subset are $2^{|Q|} = 2^3 = 8$ Here, there are 4 subsets.



DFA

DATE

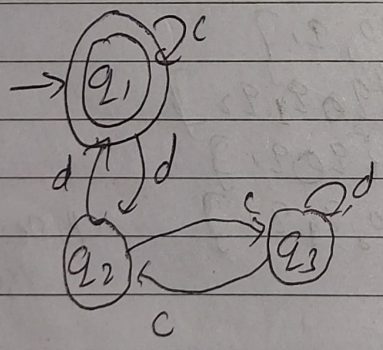
* Equivalence of NFA & DFA

The equivalence of two finite automata determines whether the two automatas are equal or not. i.e. two automatas perform the same function. The languages they accept will be same. So, in order to identify two automatas are equivalent or not, there are certain steps we need to follow:

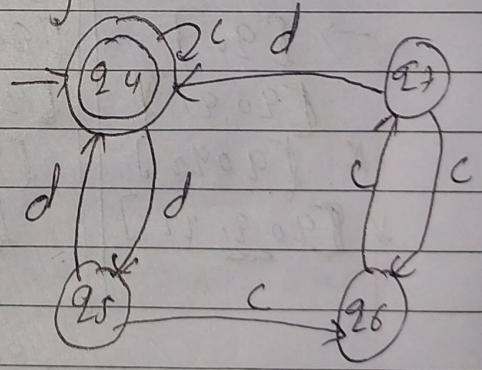
i) If initial state is final state of one automation, then in second automation also should have initial state as final state for them to be equivalent.

ii) For any pair of states $\{q_i, q_j\}$ the transition for input at Σ is defined by $\{q_a, q_b\}$ where $\delta\{q_i, a\} = q_a$ and $\delta\{q_j, a\} = q_b$. The two automata are not equivalent if for a pair $\{q_a, q_b\}$ one is intermediate state and other is final state.

Eg: Lets have two automaton namely A & B.



automata (A)



automation (B)

=) Here, in automation A, q_1 is the initial state as well as final state. & in automation B same case with q_4. So, it satisfies our first condition.

Note: Is → Intermediate state
Fs → Final state

DATE

⇒ For second condition, let's make a transition table for the pair of states from automation A & B

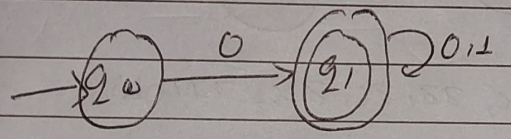
δ	c	d
(q1,q4)	(q1, q4) Fs Fs	(q2,q5) Is Is
(q2,q5)	(q3, q6) Is Is	(q1, q4) Fs Fs
(q3,q6)	(q2, q7) Is Is	(q3, q6) Is Is
(q2,q7)	(q3, q6) Is Is	(q1, q4) Fs Fs

we can see that, all the pairs checked have transition in c & d both are either initial states or final state. This satisfies the 2nd condition.

⇒ Hence, both automation A & B are equivalent.

* NFA to DFA

Q. Construct NFA for L = {set of all strings over Σ{0,1} that start with '0'} then convert it to equivalent DFA

i) 

⇒ Transition table

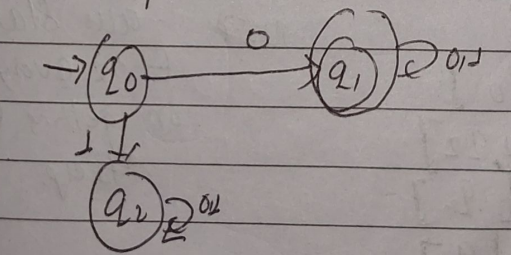
δ	0	1
→q0	q1	-
*q1	q1	q1

To convert to DFA, we 1st construct transition table for DFA

δ	0	1
→q0	q1	q2
*q1	q1	q1
q2	q2	q2

→ In DFA, transition can't be ϕ, so we made new state q2

→ Here, ϕ was changed to q2, so it's a dead state, & dead state make transition to itself



NOTE: If at least one ε symbol is seen in diagram, then we have to know that it is ε-NFA

DATE

* Epsilon NFA (ε-NFA)

An Epsilon NFA is defined as a mathematical model that consists of 5 tuple:

$E = (Q, \Sigma, \delta, q_0, F)$ where,

Q: Finite set states

Σ: finite alphabet

q_0: Initial state, $q_0 \in Q$

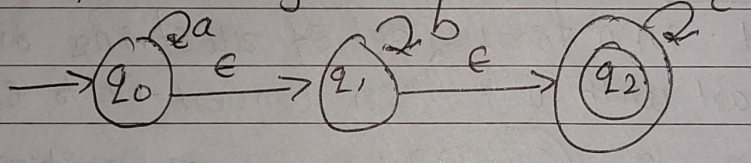
F: set of final states, $F \subseteq Q$

δ: transition function defined as $|Q| \times |\Sigma \cup \{\epsilon\}| \to 2^{|Q|}$

Ex:

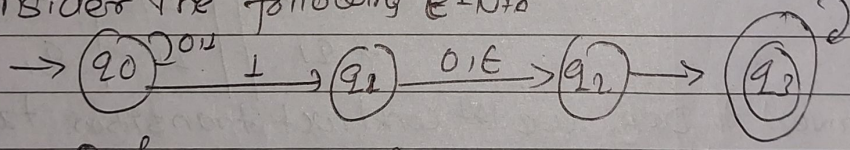
A ε-NFA that accepts following language.

L: set of strings consisting of a's followed by b's followed by c's.



$L = \{a, b, c, aa, ac, abc, aaabbbccc \ldots\}$

Ex: Consider the following ε-NFA



Now. $Q = \{q_0, q_1, q_2, q_3\}$

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

$q_0 = q_0$

$F = \{q_3\}$ & δ is defined as

δ	0	1	ε
→ q_0	$[q_0]$	$[q_0 q_1]$	$[q_0]$
q_1	$[q_2]$	ϕ	$[q_1 q_2]$
q_2	ϕ	$[q_3]$	$[q_2]$
q_3	$[q_3]$	$[q_3]$	$[q_3]$

NOTE: all states contain ε & every state on getting ε goes to its self.

DATE

* Epsilon closure (E-CLOSE)

(Informal definition): All the states reachable from a state q on reading epsilon (ϵ) only form epsilon closure of the state q.

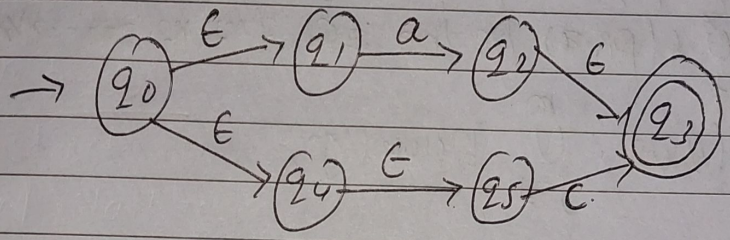
In above example,

ϵ-closure of state q_0 is $E\text{close}(q_0) = \{q_0, q_1, q_2\}$

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

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

Ex: Consider the ϵ-NFA below.



$E\text{close}(q_0) = \{q_0, q_1, q_4, q_5\}$

$E\text{close}(q_1) = \{q_1\}$

$E\text{close}(q_2) = \{q_2, q_3\}$

$E\text{close}(q_3) = \{q_3\}$

$E\text{close}(q_4) = \{q_4, q_5\}$

$E\text{close}(q_5) = \{q_5\}$

DATE

* Extended transition function of ε-NFA ($\hat{\delta}$)

Given an ε-NFA $E = (Q, \Sigma, \delta, q_0, F)$, the extended transition function for E is defined as follows to reflect what happens on a sequence of inputs.

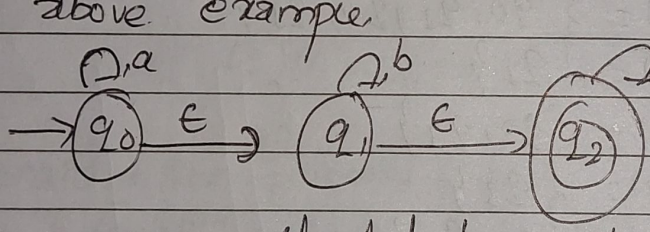
BASIS: $\hat{\delta}(q, \epsilon) = \text{Eclose}(q)$. ie. If the label of path is ε, then we can follow only ε-labeled arcs extending from state q, that is exactly what ECLOSE does

INDUCTION: Suppose w is of form xa, where a is the last symbol of w and a is a member of Σ & $a \neq \epsilon$. Then, we compute $\hat{\delta}(q, w)$ as

1. Let $\{p_1, p_2, \ldots, p_k\}$ be $\hat{\delta}(q, x)$
2. Let $\bigcup_{i=1}^{k} \delta(p_i, a)$ be the set $\{r_1, r_2, \ldots, r_m\}$

Then $\hat{\delta}(q, w) = \bigcup_{j=1}^{m} \text{Eclose}(r_j)$

In above example



Let us use extended transition function of ε-NFA to analyze the acceptance of string "aabc"

$\hat{\delta}(q_0, \epsilon) = \text{Eclose}(q_0) = \{q_0, q_1, q_2\}$

$\hat{\delta}(q_0, a) = \text{Eclose}(\delta(\hat{\delta}(q_0, \epsilon), a))$
$= \text{Eclose}(\delta(q_0, q_1, q_2), a)$
$= \text{Eclose}(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a))$
$= \text{Eclose}(\{q_0\})$
$= \{q_0, q_1, q_2\}$

DATE

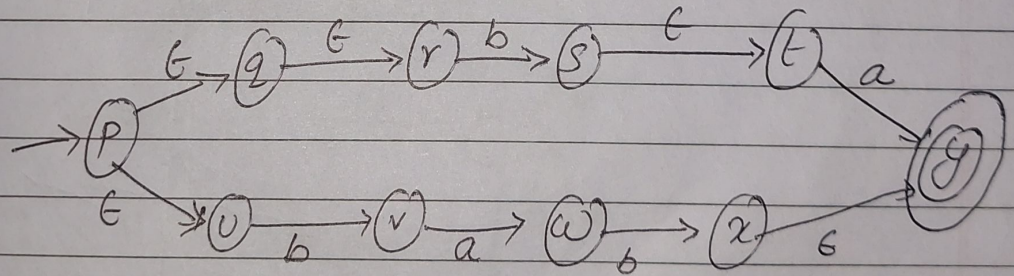
$\hat{\delta}(q_0, aa) = E\text{close}(\delta(\hat{\delta}(q_0,a),a))$
$= E\text{close}(\delta(\{q_0,q_1,q_2\},a))$
$= E\text{close}(\delta(q_0,a) \cup \delta(q_1,a) \cup \delta(q_2,a))$
$= E\text{close}(\{q_0\})$
$= \{q_0,q_1,q_2\}$

$\hat{\delta}(q_0, aab) = E\text{close}(\delta(\hat{\delta}(q_0,aa),b))$
$= E\text{close}(\delta(\{q_0,q_1,q_2\},b))$
$= E\text{close}(\delta(q_0,b) \cup \delta(q_1,b) \cup \delta(q_2,b))$
$= E\text{close}(\{q_1,q_3\})$
$= \{q_1,q_2\}$

$\hat{\delta}(q_0, aabc) = E\text{close}(\delta(\hat{\delta}(q_0,aab),c))$
$= E\text{close}(\delta(\{q_1,q_2\},c))$
$= E\text{close}(\delta(q_1,c) \cup \delta(q_2,c))$
$= E\text{close}(\{q_2\})$
$= \{q_2\}$

Since, q_2 is the accepting state, so we accept the string aabc.

Ex:



Let's check the string ba

$\hat{\delta}(p,\epsilon) = E\text{close}(p) = \{p,q,r,u\}$

$\hat{\delta}(p,\epsilon ba) = E\text{close}(\delta(\hat{\delta}(p,\epsilon),b))$
$= E\text{close}(\delta(\{p,q,r,u\},b))$
$= E\text{close}(\delta(p,b) \cup \delta(q,b) \cup \delta(r,b) \cup \delta(u,b)) = \{s,v\}$
$= \{s,t,v\}$

classmate PAGE

DATE

$\hat{\delta}(p, ba) = E\text{close}(\delta(\hat{\delta}(p, a), b))$

$= E\text{close}(\delta(\{s, t, v\}, b)$

$= E\text{close}(\delta(s, b) \cup \delta(v, b) \cup \delta(t, b))$

$= E\text{close}(\{$

$\hat{\delta}(p, ba) = E\text{close}(\delta(\hat{\delta}(p, b), a))$

$= E\text{close}[\delta(\{s, t, v, y\}, a)$

$= E\text{close}(\delta(s, a) \cup \delta(t, a) \cup \delta(v, a))$

$= E\text{close}(\{y, w\})$

$= \{y, w\}$

Here, the final result set i.e. $\{y, w\}$ contains one of the final state y. So, the string ba is accepted.

DATE

Some examples to convert NFA to DFA.

Ex. Find the equivalent DFA for the NFA given by $M = [\{A, B, C\}, (a, b), \delta, A, \{C\}]$ when δ is given by

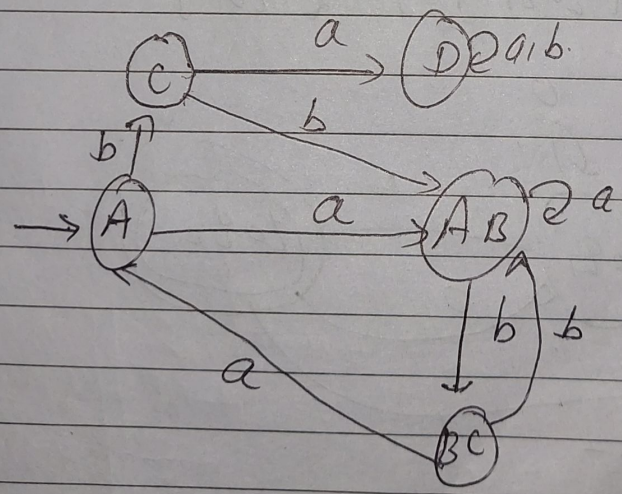
δ	a	b
→ A	AB	C
B	A	B
* C	$\emptyset$	AB

⇒ For converting into equivalent DFA, Let's draw the required transition table for DFA.

δ	a	b
→ A	AB	C
→ AB	AB	BC
* C	D	AB
* BC	A	AB
D	D	D

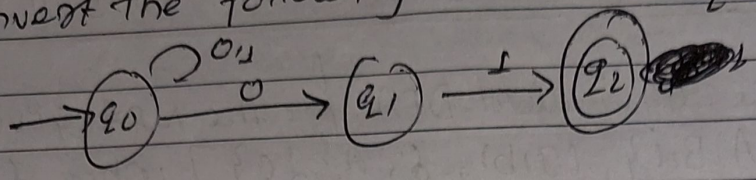
→ DFA can't have $\emptyset$, so we make a new state D, which is a dead state.

← since D is dead state.



DATE

Ex. Convert the following NFA to equivalent DFA

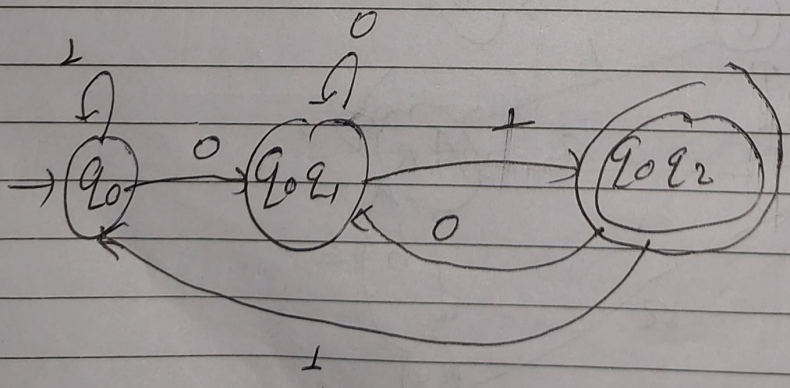


Let's make the transition table

δ	0	1
→q0	q0q1	q0
q1	ϕ	q2
*q2	ϕ	ϕ

Let's make transition table for DFA

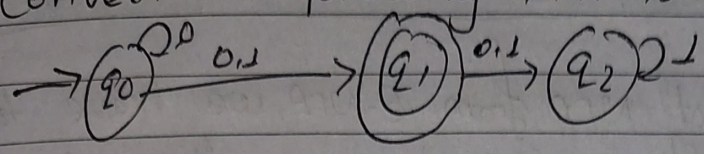
δ	0	1
→q0	{q0q1}	{q0}
q0q1	{q0q1}	{q0q2}
*q0q2	{q0q1}	{q0}



DATE

* ~~Minimization of DFA~~

Q. Convert the following NFA to its equivalent DFA

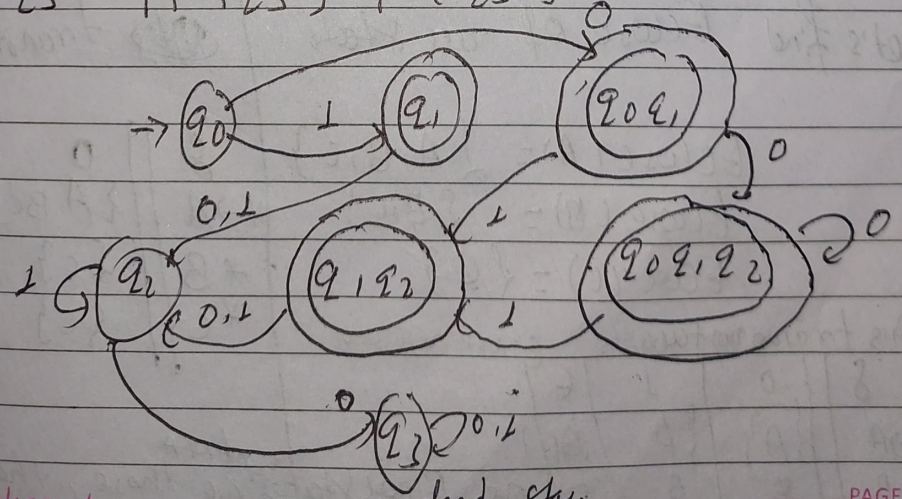


=) Let's make transition table 1st

δ	0	1
→ q_0	$\{q_0 q_1\}$	$\{q_1\}$
* q_1	$\{q_2\}$	$\{q_2\}$
q_2	ϕ	$\{q_2\}$

Lets make transition table for DFA

δ	0	1
→ q_0	$\{q_0 q_1\}$	$\{q_1\}$
* $q_0 q_1$	$\{q_0 q_1 q_2\}$	$\{q_1 q_2\}$
* q_1	$\{q_2\}$	$\{q_2\}$
* $q_0 q_1 q_2$	$\{q_0 q_1 q_2\}$	$\{q_1 q_2\}$
* $q_1 q_2$	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$
q_3	$\{q_3\}$	$\{q_3\}$



DATE

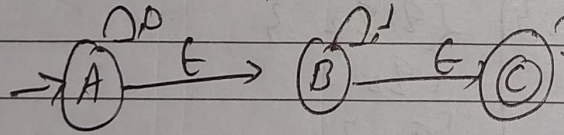
* Removing Epsilon Transition using the concept of Epsilon closure.

(Converting ε-NFA to its equivalent NFA)

To construct NFA from ε-NFA, we have to eliminate the ε-transitions somehow, so that the resulting NFA will have no more ε-transitions. For this we do as below:

i) For each state, we find the ε closure
ii) Now set of states that we get in (i) have to be checked on which state they make transition on getting a particular input.
iii) Now, again find the ε closure of all the states we get in (ii)

Ex. Convert the following ε-NFA to NFA



→ Let's find ε closure of all states

ε closure (A) = {A, B, C}
ε closure (B) = {B, C}
ε closure (C) = {C}

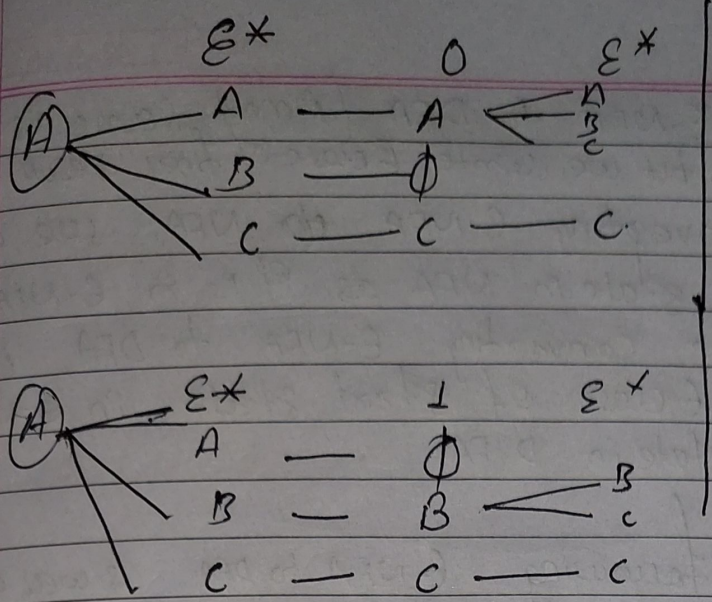
This transition table is not necessary

δ	0	1	ε
→A	{A}	{B}	{B}
B	ϕ	B	{B, C}
*C	C	C	C

NFA transition table

δ	0	1
*→A	{A, B, C}	{B, C}
*B	{C}	{B, C}
*C	{C}	{}

Note: Final states of NFA are those, that can reach final state only by reading ε



DATE

or $\delta(A, 0)$

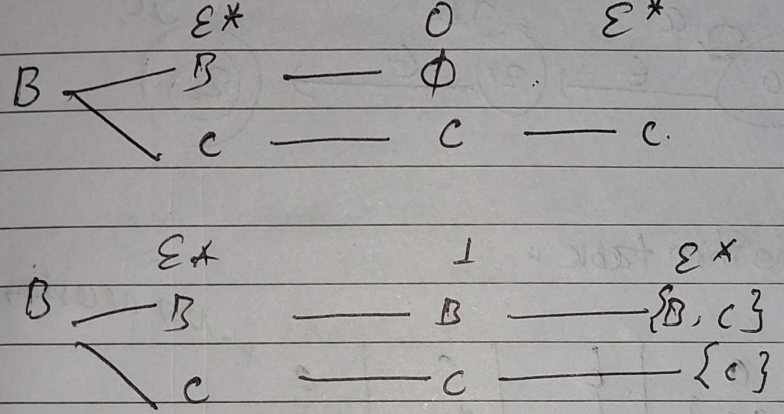
$= \epsilon\text{-closure}(\delta(\epsilon\text{closure}(A), 0))$

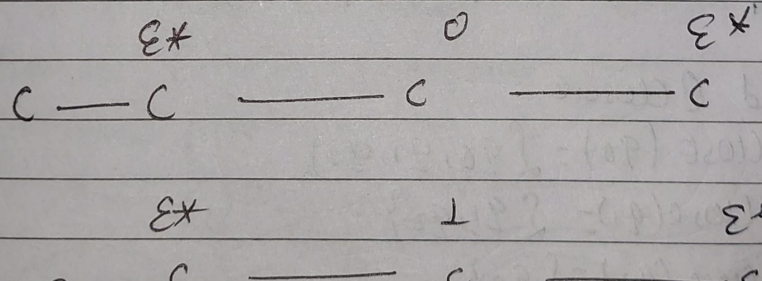
$= \epsilon\text{-closure}(\delta(\epsilon\text{closure}(A), 0))$

$= \epsilon\text{closure}(\delta\{A, B, C\}, 0)$

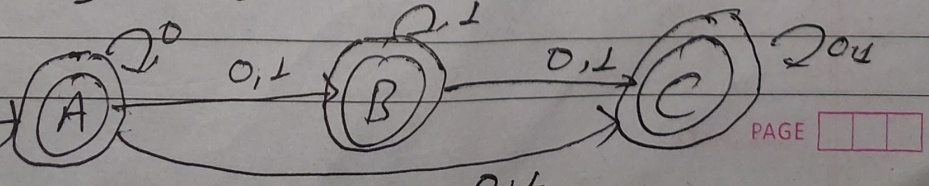
$= \epsilon\text{closure}(A, C)$

$= \{A, B, C\}$





It's equivalent NFA =



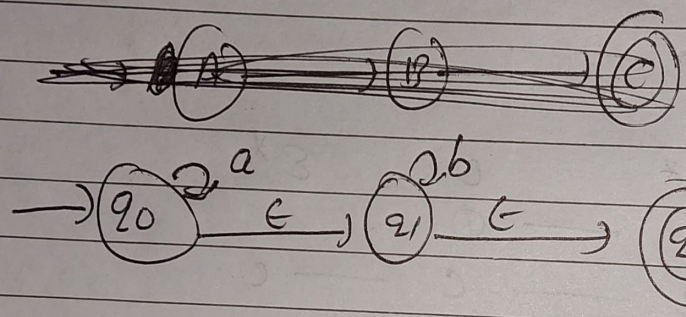
classmate

PAGE

DATE

* Converting ϵ-NFA to DFA. (Almost same as NFA to DFA, but we write ϵ close of final result. While converting ϵ-NFA to NFA, we used same start state in NFA as it is in ϵ-NFA but now for converting ϵ-NFA to DFA, we will use ϵ close of start state in ϵ-NFA as a start state in DFA.

Eg: Convert following ϵ-NFA to DFA as well as NFA



⇒ It's transition table :

δ	a	b	c	ϵ
→ q0	{q0}	ϕ	ϕ	{q1}
q1	ϕ	{q1}	ϕ	{q2}
* q2	ϕ	ϕ	{q2}	{q2}

← Not necessary

Let's find ϵ closure.

ϵ-CLOSE (q0) = {q0, q1, q2}

ϵ-close(q1) = {q1, q2}

ϵ-close (q2) = {q2}

DATE

Now, let's make transition table for DFA

δ_D	a	b	c
→ $q_0q_1q_2$	$\{q_0q_1q_2\}$	$\{q_1q_2\}$	$\{q_2\}$
* q_1q_2	ϕ or q_3	$\{q_1q_2\}$	$\{q_2\}$
* q_2	ϕ or q_3	ϕ or q_3	q_2
q_3	q_3	q_3	q_3

Note: $\delta(\{q_0,q_1,q_2\}, a) = \text{E-close}(\delta\{q_0,q_1,q_2\}, a)$

$= \text{E-close}(\delta(q_0,a) \cup \delta(q_1,a) \cup \delta(q_2,a))$

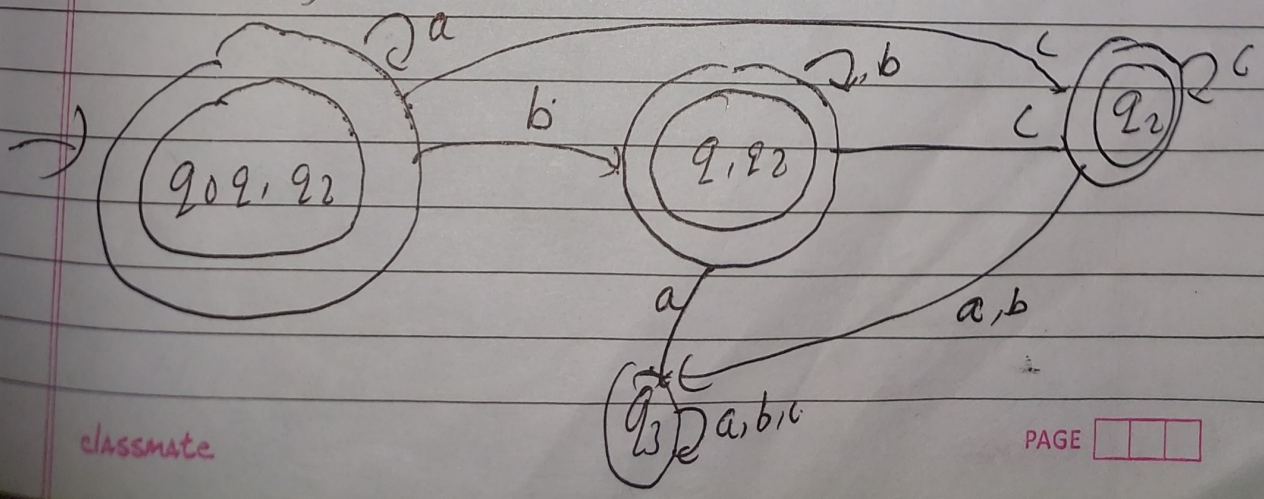
$= \text{E-close}(\{q_0\})$

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

in 1st row, 2nd column.

Or Here, if we want to find $\delta(\{q_0,q_1,q_2\}, b)$ then see transition table of E-NFA there you can see. Its value is $\{q_1\}$ only, Now again find the E-close of q_1 and write here

Its start state is the E-close of start state of E-NFA, and final state is all the states containing final state of E-NFA



classmate

PAGE

DATE

To Convert into NFA

→ same in both ENFA to NFA × ENFA to DFA

δ	a	b	c
→q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
*q_2	ϕ	$\{\phi\}$	$\{q_2\}$

different in both

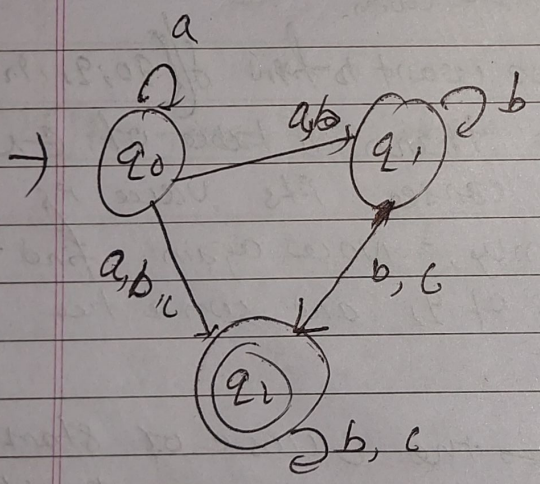
Now $\delta(q_0, a) =$ ~~Eclose(Eclose(δ(q~~

$= E\text{close}(\delta(E\text{close}(q_0), a)$

$= E\text{close}(\delta(q_0, q_1, q_2), a)$

$= E\text{close}(\{q_0\})$

$= \{q_0 q_1 q_2\}$



Note: Remember, ENFA to NFA लैजाँदा aja adi δ को तेहा states haru same as ENFA

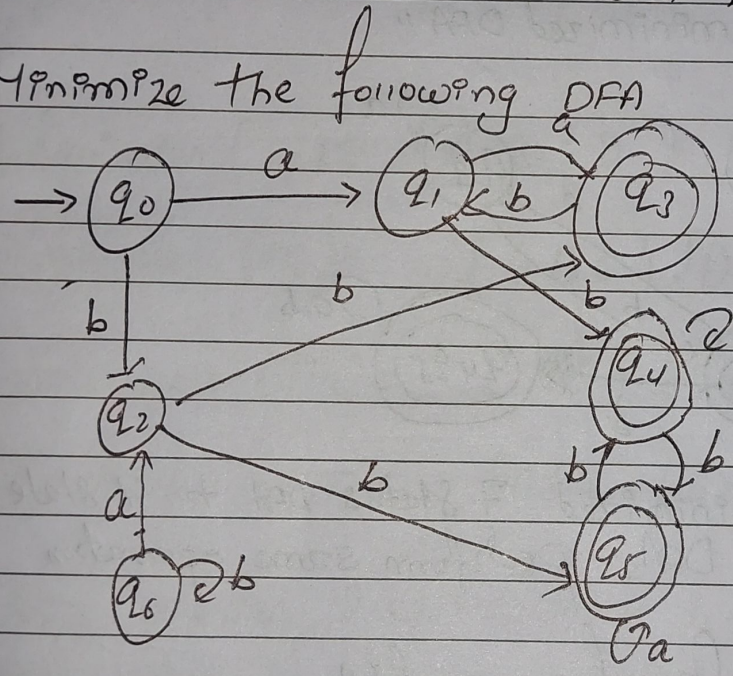
ENFA to DFA लैजाँदा δ को state ε-closure of states an same ho dui tai ta.

Note: Two states p, q are equivalent if $\delta(p,w) \in F \Rightarrow \delta(q,w) \in F$ or $\delta(p,w) \notin F \Rightarrow \delta(q,w) \notin F$

* Minimization of DFA : Partitioning Approach:

Minimization of DFA is required to obtain the minimal version of any DFA which consists of the minimum number of states possible. DFA with less states will perform better than the one with more states. Hence, minimization of DFA is required

Ex: Minimize the following DFA



Let's make transition table by removing unreachable state which is q_6.

δ	a	b
→ q_0	q_1	q_2
q_1	q_3	q_4
q_2	q_3	q_5
* q_3	q_3	q_1
* q_4	q_0	q_5
* q_5	q_5	q_4

Now, based on the transition table, lets make the equivalence class of accepting and non accepting state.

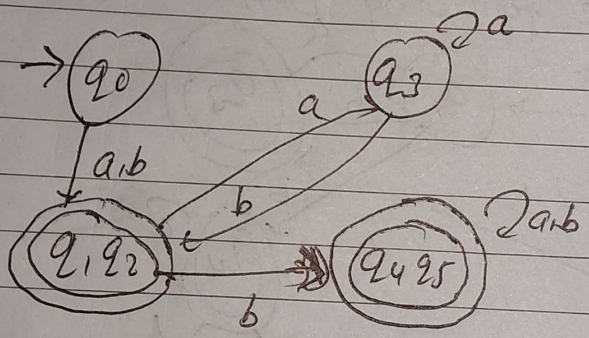
0-Equivalent (π_0) => { q_0 q_1 q_2 } { q_3 q_4 q_5 }

1-Equivalent (π_1) => { q_0 } { q_1 q_2 } { q_3 } { q_4 q_5 }

2-Equivalent (π_2) => { q_0 } { q_1 q_2 } { q_3 } { q_4 q_5 }

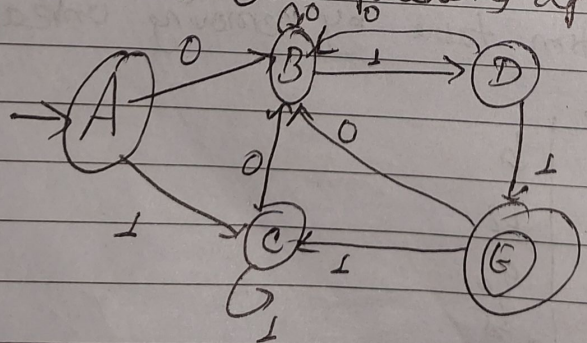
Here, 1-Equivalent & 2-Equivalent give same result.

Now, minimized DFA is



Here, we minimized 7 states DFA to 4 state DFA where both DFAs perform same operation.

Ex-2. Minimize the following dfa



DATE

Let's make the transition table removing the unreachable states.

δ	0	1
→A	B	C
B	B	D
C	B	C
D	B	E
*E	B	C

Now, based on the transition table, lets make the equivalence class of accepting & non accepting stat

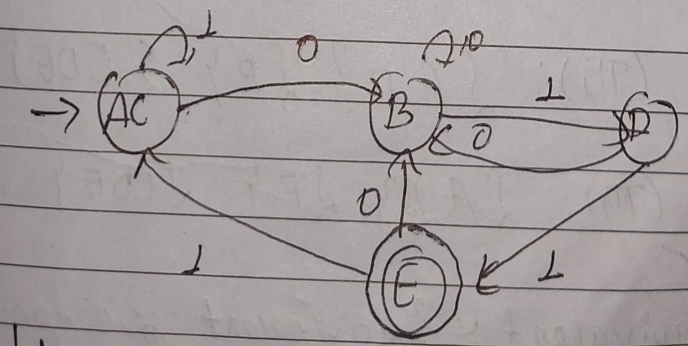
0-Equivalent (π_0) = {A B C D} {E}

1-Equivalent (π_1) = {A B C} {D} {E}

2-Equivalent (π_2) = {A C} {B} {D} {E}

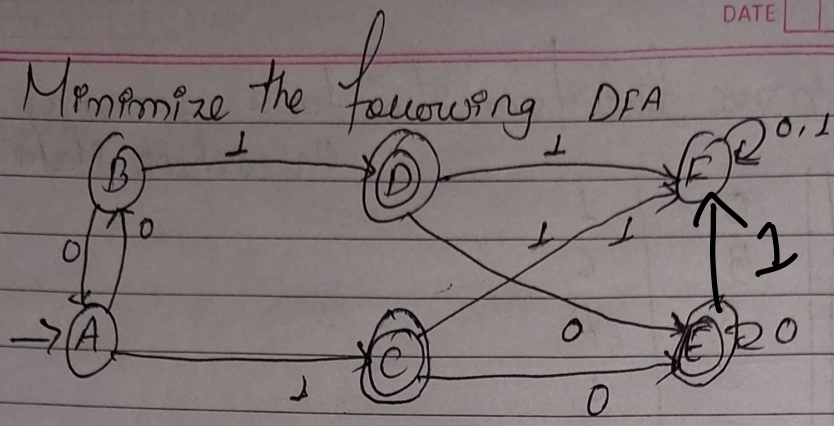
3-Equivalent (π_3) = {A C} {B} {D} {E}

Now, we see 2-equivalent & 3-equivalent give same result. So, minimized DFA is (Note: you can also make transition table for new dfa)



Here, we minimized 5 state DFA to 4 state DFA where both DFAs perform same operation

Ex: Minimize the following DFA



⇒ Let's make the transition table 1st removing the unreachable states.

δ	0	1
→ A	B	C
B	A	D
* C	E	F
* D	E	F
* E	E	F
F	F	F

Now, based on the transition table, let's make the equivalence class of accepting & non accepting state.

0-Equivalent (π_0): {ABF} {CDE}

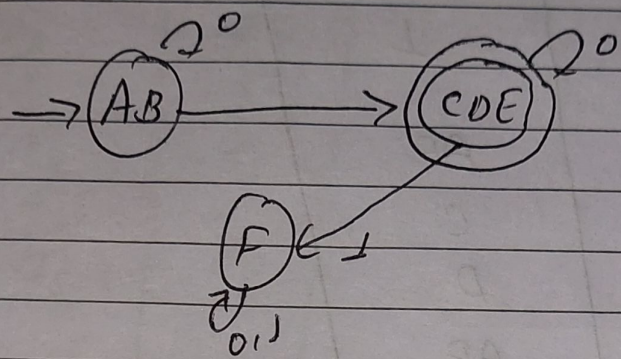
1-Equivalent (π_1): {AB} {F} {CDE}

2-Equivalent (π_2): {AB} {F} {CDE}

We see 2-Equivalent & 3-Equivalent give same result. So, the transition table for minimised DFA is:

DATE

	δ	0	1
→	AB	{AB}	{CDE}
	F	{F}	{F}
*	CDE	{CDE}	{F}



Ex: Find the minimum state DFA for the given DFA below.

States	Inputs	
	0	1
→A	B	F
B	E	C
C	B	D
* D	E	F
E	B	C
F	B	A

⇒ Based on the transition table, let's make the equivalence class of accepting & non-accepting

0-Equivalent (Π_0): {A B C E F} {D}

1-Equivalent (Π_1): {A B E F} {C} {D}

2-Equivalent (Π_2): ~~{A B E F}~~ {C} {D}
{A F} {B E}

PAGE

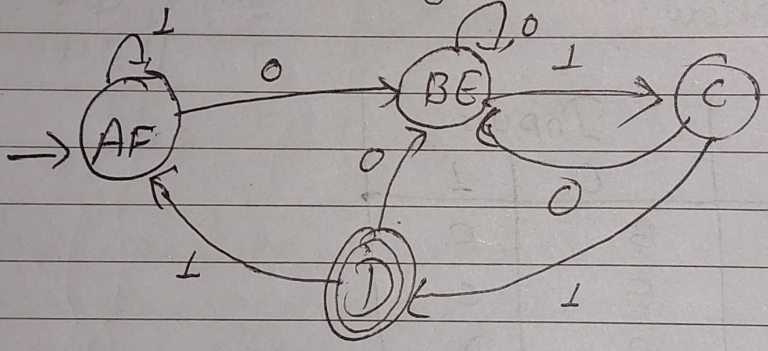
DATE

3-Equivalent (T_3): {AF} {BE} {C} {D}

Let's make transition table for minimized DFA

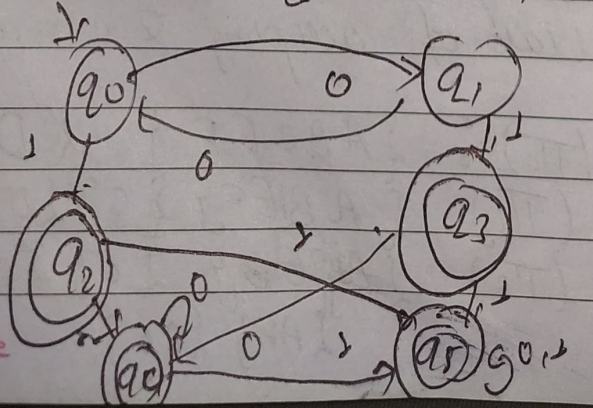
δ	0	1
→ AF	BE	AF
BE	BE	C
C	BE	D
* D	BE	AF

DFA transition state diagram is



Hence, we minimized the DFA from 6 states to 4 states

Q. ~~Minimize~~ Minimize the following



DATE

Let's make transition table eliminating the unreachable states:

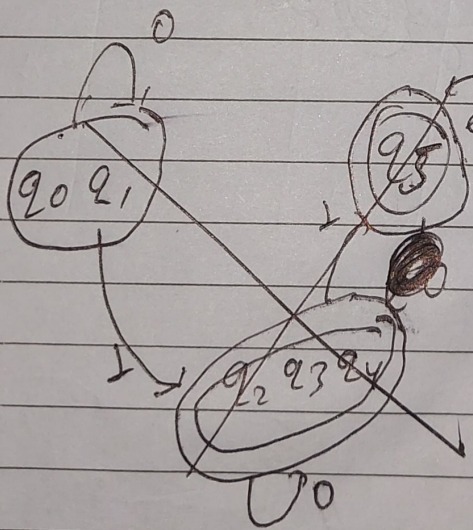
	δ	0	1
→	q_0	q_1	q_2
	q_1	q_0	q_3
*	q_2	q_4	q_5
*	q_3	q_4	q_5
*	q_4	q_4	q_5
	q_5	q_5	q_5

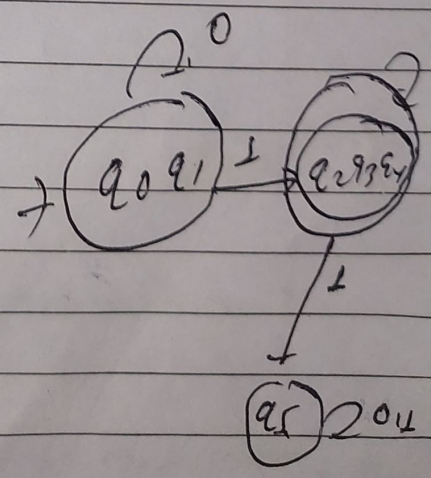
Let's make the equivalence class

0-equivalent (π_0): $\{q_0 q_1 q_5\}$ $\{q_2 q_3 q_4\}$

1-equivalent (π_1): $\{q_0 q_1\}$ $\{q_5\}$ $\{q_2 q_3 q_4\}$

2-equivalent (π_2): $\{q_0 q_1\}$ $\{q_5\}$ $\{q_2 q_3 q_4\}$





$L = \{101, 001, 00100, \ldots\}$

$\therefore R: 0^*10^*$

classmate

PAGE

DATE

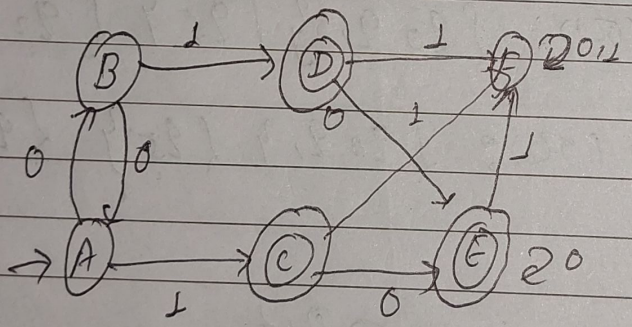
* DFA Minimization: Table Filling Method (Myhill Nerode Theorem)

Steps:

1. Draw a table for all pairs of states (P, Q)
2. Mark all pairs where $P \in F$ and $Q \notin F$
3. If there are any unmarked pairs (P, Q) such that $[\delta(P, x), \delta(Q, x)]$ is marked, then mark $[P, Q]$, where x is an input symbol.

   REPEAT THIS UNTIL NO MORE MARKINGS CAN BE MADE.
4. Combine all the unmuted pairs and make them a single state in the minimized DFA.

Ex:



PAGE

Finite state machines with output:

DATE

1. Moore Machine.

Moore machine is a finite state automata with output consisting of 6 tuples.

$M_0 = \{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$

where,

Q = Finite set of state

Σ = input symbol (alphabet)

δ = Transition function $\delta: Q \times \Sigma \to Q$

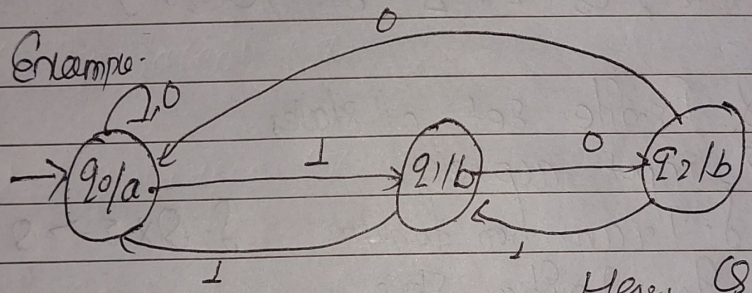
q_0 = start state.

Δ = output symbol.

λ = output function.

$\lambda: Q \to \Delta$

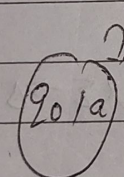
Example:



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

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

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



NOTE: q_0/a means, state q_0 on getting input 0 goes to itself without output a.

$\lambda: Q \to \Delta \Rightarrow \begin{matrix} q_0 \to a \\ q_1 \to b \\ q_2 \to b \end{matrix}$

Note: If you are giving input of n length string, it gives output of n+1 length.

Note: There is no concept of final states in finite state machines with known output.

PAGE

DATE

Transition table:

Current state	Next state		Output
	0	1	
→ q_0	q_0	q_1	a
q_1	q_2	q_0	b
q_2	q_0	q_1	b

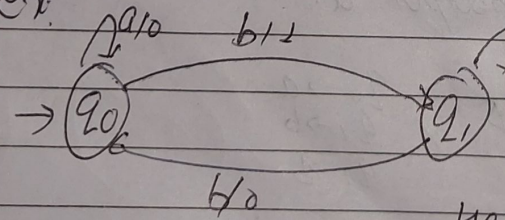
~~Mealy~~ Mealy Machine

It is a finite state automata with output consisting of 6 tuples:

$$Me: \{ Q, \Sigma, \Delta, \delta, \lambda, q_0 \}$$

Q: Finite set of states
Σ: input alphabet
δ: Transition function: $\delta: Q \times \Sigma \to Q$
q_0: Initial state
Δ: Output symbol
λ: Output function
$\lambda: Q \times \Sigma \to \Delta$

Ex.



Here, (a/b) → input symbol
(0/1) → output symbol

λ: $q_0, a \to 0$
$q_0\ b \to 1$
$q_1, a \to 1$
$q_1, b \to 0$

DATE

Note: If you give input string of length n, it gives output of same length n.

Transition table:

Current state	Next state			
	Input (a)		Input (b)	
	state	output	state	output
→ q_0	q_0	0	q_1	1
q_1	q_1	1	q_0	0

* Difference between Moore & Mealy Machine

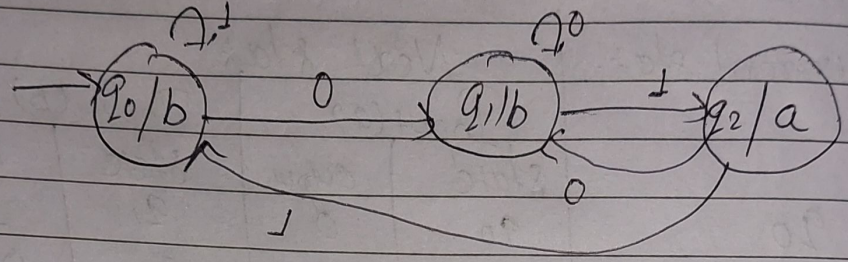
Moore machine	Mealy Machine
i) Output only depends on present state	i) Output depends on present state & present input
ii) Input string length 'n' → output string 'n+1'	ii) Input string of length 'n' → output also 'n' length
iii) Output is synchronous with clock.	iii) Output is asynchronous
iv) Generally it has more states than Mealy.	iv) Generally, it has fewer states than Moore machine.
v) Draw example.	v) " "

PAGE

DATE

Ex: Construction of Moore Machine

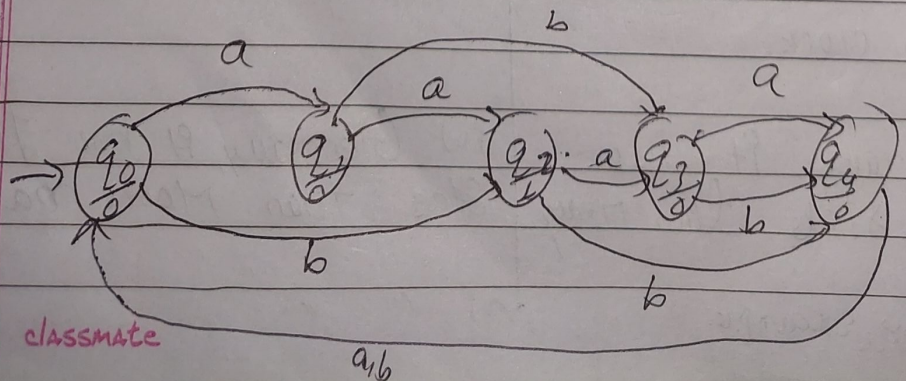
Ex: Construct a Moore Machine that prints 'a' whenever the sequence '01' is encountered in any input binary string



Ex: For the following Moore Machine the input alphabet is $\Sigma = \{a, b\}$ & the output alphabet is $\Delta = \{0, 1\}$. Run the following input sequences & find the respective outputs.

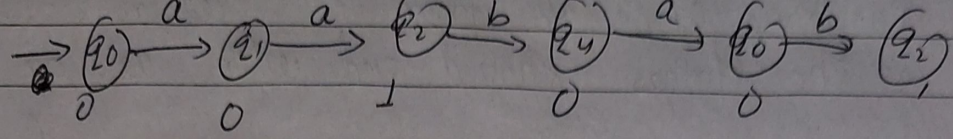
i) aabab ii) abbb iii) ababb

State	a	b	output
→ q_0	q_1	q_2	0
q_1	q_2	q_3	0
q_2	q_3	q_4	1
q_3	q_4	q_4	0
q_4	q_0	q_0	



DATE

Soln i) aabab

=> 

Hence, output is: 00100 1

ii) abbb

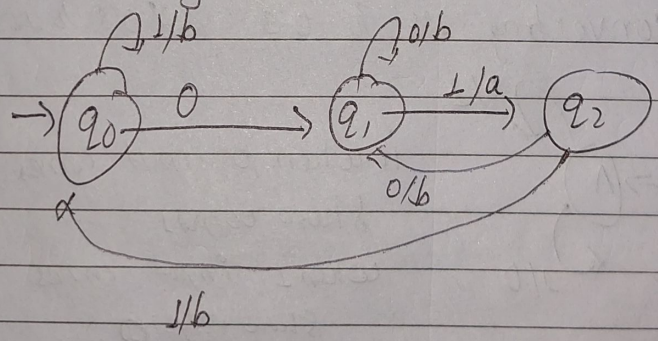
=> output is: 0000

iii) ababb

=> output is 00001

Ex: Construction of Mealy Machine.

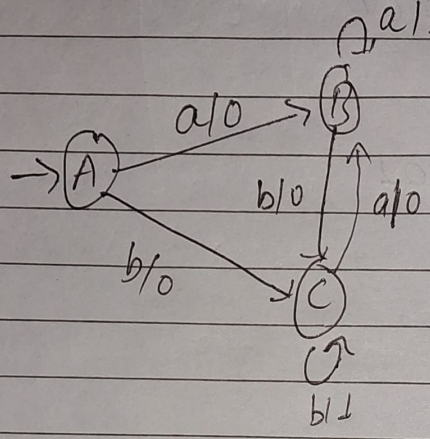
Ex: Construct a Mealy machine that prints 'a' Whenever the sequence '01' is encountered in any input binary string



DATE

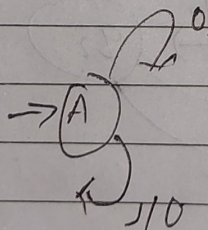
Ex: Design a Mealy machine accepting the language consisting of strings from Σ^*, where $\Sigma = \{a, b\}$. & the strings should end with aa or bb

=> Σ^* means all any length of strings from $\Sigma = \{a, b\}$. Our string should end with aa or bb. So, let if we get sequence aa or bb then we print output as 1. o/w 0.



Ex: Construct a Mealy machine that produces the 1's Complement of any binary input string

=> We know that 1's Complement of any binary string is converting 1's to 0's & 0's to 1's.



ie. when 0 input comes show output 1
when 1 input comes show output 0

Questions asked from this Chapter
(In Board Exam)

DATE

Q. Give the formal definition of DFA and NFA. How NFA can be converted into equivalent DFA? Explain with suitable example. (10 marks, 2078)

Q. Find the minimum state DFA for the given DFA below. (10 marks - 2078)

States	Inputs	
	0	1
→A	B	F
B	E	C
C	B	D
*D	E	F
E	B	C
F	B	A

Q. Define the NFA with ϵ-Transition & ϵ-closure of a state. Show that for every regular expression r, representing a language L, there is ϵ-NFA accepting the same language. Also convert regular expression (a+b)* ab* into equivalent finite Automata. (10 marks, 2+6+2, 2076)

Q Give the formal definition of DFA. Construct a DFA accepting all strings of {0,1} with even number of 0's and even number of 1's (5 marks - 2076)

Q. Construct a NFA accepting language of {0,1} with each string ending with 01 & convert it into equivalent DFA. (2076, 5 marks)

CLASSMATE

PAGE

DATE

Q. How a ε-NFA can be converted into NFA and DFA? Explain with a suitable example. (2067-5 marks) (2071-5 marks)

Q. Explain about subset construction method to convert a NFA into equivalent DFA with suitable example. (2076-5 marks) (2072-5 marks)

Q. Give formal notation for an ε-NFA with example. (2075-5 marks) (2069-5 marks) (2073-5 marks) (2074-5 marks)

Q. Construct a DFA that accepts the strings over alphabet {0,1} with odd number of 0's and even number of 1's. (2075-5 marks)

Q. " " " " " " even number of 0's & even no. of 1's (2071-5 marks)

Q. Construct a NFA accepting the language over {a,b} with each strings containing three consecutive b's. Show by extended function if it accepts abbb (2074-5 marks)