Theory of Computation

Symbols :- Smallest entities, that cann't be broken down further, they are the characters, which are used to create strings/words of a language. e.g. '0', '1', '2', '3', ... are symbols of Math language.

Alphabets :- An alphabet is a finite, non-empty set of symbols. The alphabet of a language is usually represented by Σ.

e.g. $\Sigma = \{0, 1\}$, $\Sigma = \{a, b, c, d\}$, $\Sigma = \{x, +, -, /\}$

Strings :- A string or word is a finite sequence of symbols from an alphabet set.

e.g. 10 is a string from $\Sigma = \{0, 1\}$

similarly 1, 0, 11, 00, 100 all are strings over $\Sigma = \{0, 1\}$

* a string could be of zero length as well, they are known as empty strings, and they are represented as ϵ.
* length of a string is the number of symbols in the string, it is represented by $|w|$, where w is the notation for word or string.

$|101| = 3$, $|01| = 2$, $|1| = 1$, $|0000| = 4$

Language :- A language over an alphabet is a set of strings over the alphabet. Over $\Sigma = \{a, b, c\}$, Any no. of languages could be defined. e.g.

$L_1 = \{aa, a, bb, cc, ab\}$

$L_2 = \{a, b\}$

$L_3 = \{abc, cba\}$

$L_4 = \{a, aa, ab, ac, abc, aba\}$

$L_5 = \{a, b, c, ab, ac, aa, ba, \dots\}$

It can be finite as well as infinite. A language can be empty as well, if it doesn't contain any string, it is represented using empty brackets or ϕ. A language containing empty string is not an empty language. One has Zero cardinality and other has one.

Operations on Strings :-

1.) Concatenation :- Two strings can be concatenated is when one string is appended after another. e.g. if 'ab' and 'ac' are two strings, then their concatenation operation is represented using . operator, i.e. $ab.ac = abac$, i.e. it produces a new string 'abac' if we concatenate ab after ac then it generates $ac.ab = acab$. So concatenation is not commutative.

2. Reversal :- For any string $w = a_1 a_2 a_3 \dots a_n$, the reversal of the string will be $w^R = a_n a_{n-1} \dots a_2 a_1$.
e.g. if $w = abc$ then $w^R = cba$.
if $w = aa$ then $w^R = aa$.

3. Prefix :- Collection of all the strings that act as prefix of the given string. Every strings is its own prefix, also null string is prefix of every other string. e.g. for $w = 0110$ the prefixes are : $\epsilon, 0, 01, 011, 0110$.

ϵ 1 2 3 ϵ

4.

4. Suffixe :- Collection of all strings that act as suffixe of the given string. Every string acts as its own suffix. Null string is also suffix of every

classmate
Date
Page

String : e.g. for $w = abcb$,
suffixes are : $\epsilon, abcb, b, cb, bcb$.

Substrings :- Collection of all parts/subparts of the string is known as Substrings. eng. for Every string is its own substring, null string is also substring of every string. e.g. for $w =$ 'abac', substrings are a, b, c, ab, ba, ac, aba, bac, abac, ϵ.

Power of a string :- power of string means concatenating a string with itself the same number of times as the power. e.g $(ab)^2 = ab.ab = abab$.
$(aca)^3 = aca.aca.aca = acaacaaca$.

Power of Alphabet :- Σ^n means collection of all strings of length 'n' with symbols of Σ.
e.g. if $\Sigma = \{a, b, c\}$
then, $\Sigma^1 = \{a, b, c\}$
$\Sigma^2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$
$\Sigma^3 = \{aaa, aab, aac, aba, abb, abc, ----\}$
$\Sigma^0 = \{\epsilon\}$
The set of all strings over Σ is denoted by Σ^*, known as Kleene closure. $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup -- \Sigma^n ...$

Positive closure : The set of all strings over Σ, which contains atleast one symbol is represented as Σ^+. $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup ---- \Sigma^n ...$

$$\Sigma^* = \Sigma^0 \cup \Sigma^+$$

Σ^* & Σ^+ are always infinite. Any language L is always a subset of Σ^*.

Finite Automata

An automaton is defined as a system that performs certain functions without human intervation. It can be designed to perform a variety of tasks related to various domains of human life, but over here we are focusing on the language recognition process.

The simplest kind of language recognition machine is the finite state machine (FSM) or also known as finite automata (FA). Finite automata can recognize Regular language.

A FSM consists of a number of states and some basic rules for transition from one to another. FSM have an input tape, which contains string of characters, each placed in a cell of input tape.

A string is recognized by the FSM if on the complete consumption of the string (input tape), it halts in an acceptor or final state. If machine is not in final state after processing of the string, then we say string is rejected implying it doesn't belong to the given language.

A finite automata could be of two types based upon its nature:

1.) Deterministic Finite Automata (DFA)
2.) Non-deterministic Finite Automata (NFA)

Both the DFA & NFA has the same power, meaning every machine that exists in DFA form, can be constructed in NFA form, and vice-versa.

DFA or deterministic finite automata, is a machine in which we are always certain how the machine is going to behave, whereas in case of NFA, we are aware how the machine will work for the strings that belongs to the language, for the rest we donot care, how the machine will work.

Conditions of Determinism:

1.) On any state we cannot have multiple outgoing transitions with the same symbol.
2.) On every state we must have a transition for every input alphabet.

Formal definition of Finite Automata :

A finite automata is a five-tuple structure $M(Q, \Sigma, q_0, \delta, F)$, where

1.) Q is a finite set of states in which the finite automata can exist.
2.) Σ is set of input symbols.
3.) q_0 is the starting state of the finite automata.
4.) δ is the transition function that determine the next state, given the current state of the finite automata & the current input symbol.
5.) $F \subseteq Q$ is the set of final states.

A DFA is represented with the help of these 5 tuples :

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

where M is the name of DFA.

δ, Transition function of DFA is a tedious work to write, & is heard to understand, so the preferred ways for representing δ are :

1.) Transition diagram/graph
2) Transition table.

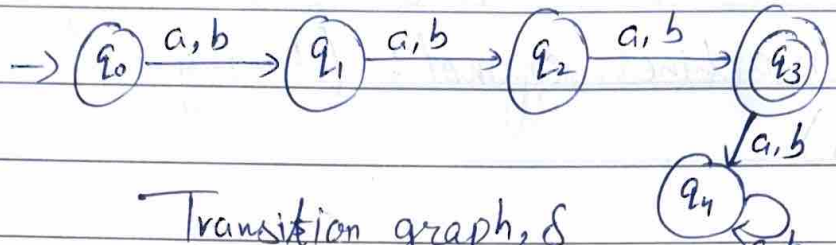
Transition Graph: A transition graph for $M = (Q, \Sigma, \delta, q_0, F)$ is a graph defined as follows:

1) For each state in Q, there is a node.
2) For each state q in Q and each input symbol a in Σ, then if $\delta(q, a) = q_1$. Then the graph will have an edge from q to q_1, labelled as 'a'.
3) Start state q_0 will be represented by an arrow. This arrow does not originate at any node.
4) The final states in F, are represented by double circles, for rest we use only single circle.

Transition Table: It is a conventional tabular representation of function δ. The rows of the table corresponds to the states, and the columns corresponds to the inputs. The entry for row corresponding to state q and the column corresponding to input a, is the state $\delta(q, a)$

Q. Design a DFA, that will accept all the strings of length 3, defined over $\Sigma = \{a, b\}$.

Sol.



Transition graph, δ

Transition table :

	a	b
→ q_0	q_1	q_1
q_1	q_2	q_2
q_2	q_3	q_3
q_3	q_4	q_4
q_4	q_4	q_4

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

where $Q = \{q_0, q_1, q_2, q_3, q_4\}$

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

δ : $\delta(q_0, a) = q_1$ $\quad \delta(q_1, a) = q_2$

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

and so on, or otherwise, we can show it with help of above transition table or the ~~fun~~ graph.

$q_0 = q_0$, initial state

$F = \{q_3\}$, final state

String processing by DFA:

Let's say we have input aabc, & we have to check whether it belong to previous machine or not: (having exactly length 3).

$$\delta(q_0, aabc) = \delta(q_1, abc) = \delta(q_2, bc)$$

$$= \delta(q_3, c) = q_4$$, since q_4 is a non-final state, so this string doesnot belong to the language.

Specific States:

1.) Dead state: A state is a dead state if we cannot reach final state from dead state, by taking any number of transistions, e.g. q_4 in previous automata.

2.) Unreachable State: A state is said to be unreachable if we cannot reach the state starting from initial state by taking any no. of transistions.

* Dead state is the only unproductive state which cannot be removed from the system, otherwise it will make system incomplete.

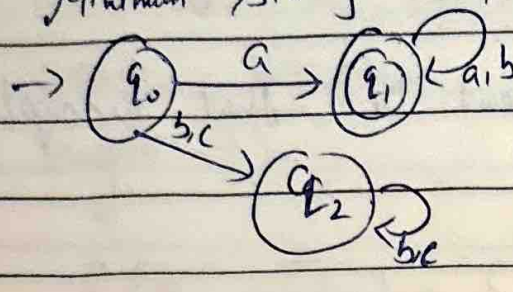
* We can have two different DFA that accepts the same language.

* Minimal DFA's are always unique & we cannot have two different minimal DFAs, that accepts the same language.

Design a DFA. for the following languages:

1.) $L = \{w \mid w \in \Sigma^*, \Sigma = \{a, b, c\}, w \text{ starts with } a\}$

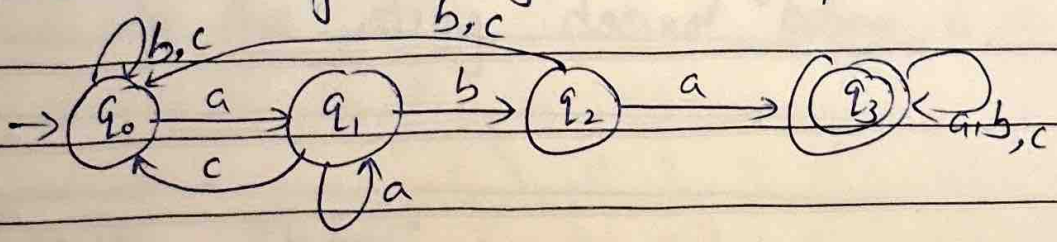
Minimum string acceptable is $\underline{a}$.



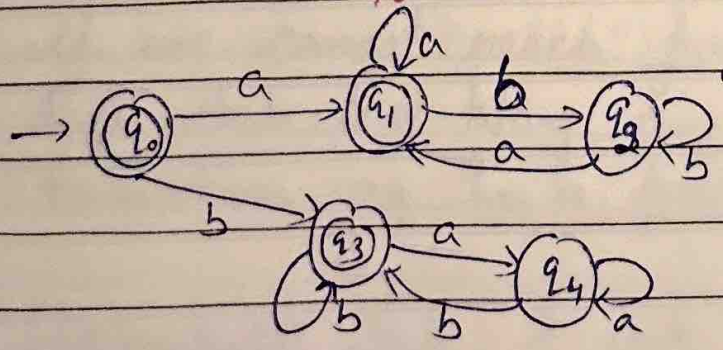
q_0 is the initial state
q_1 is the final state
q_2 is the dead state

2.) $L = \{w \mid w \in \Sigma^*, \Sigma = \{a, b, c\}, aba \text{ is a substring of } w\}$.

Minimum length string to be accepted is aba.



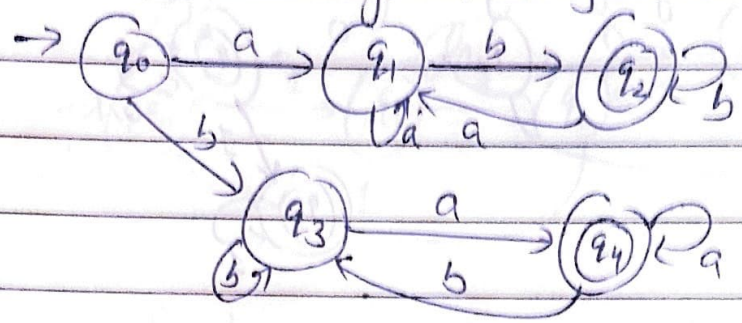
3.) $L = \{w \mid w \in \Sigma^*, \Sigma = \{a, b\}, w \text{ starts \& ends with same character}\}$



Minimum length string = ϵ, a, b

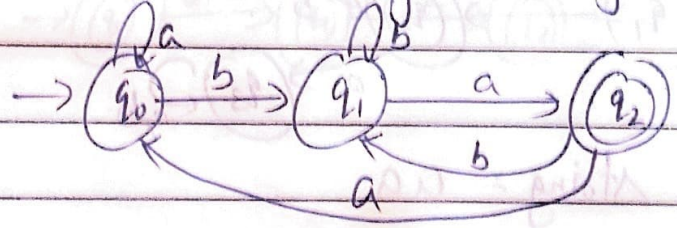
4) $L = \{w \mid w \in \Sigma^*, \Sigma = \{a,b\}$, it starts & ends with different characters$\}$

Minimum length string = ab, ba



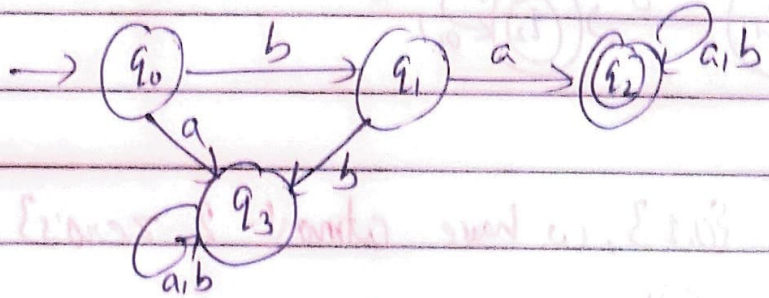
5) $L = \{w \mid w \in \Sigma^*, \Sigma = \{a,b\}$, w ends with ba$\}$

Minimum length string = ba



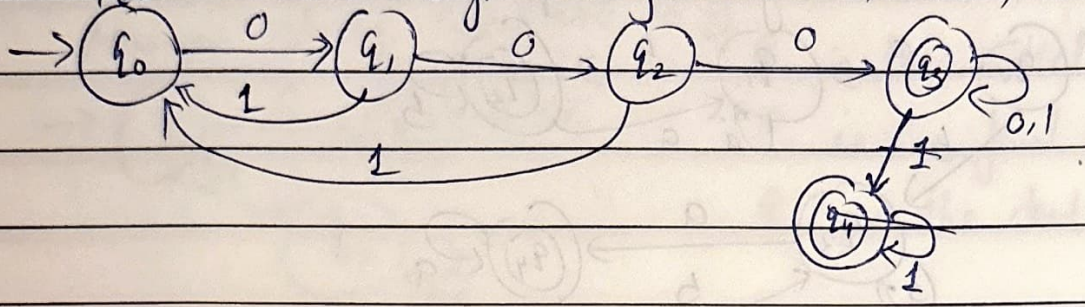
6) $L = \{w \mid w \in \Sigma^*, \Sigma = \{a,b\}$, w starts with ba$\}$

Minimum length string = ba

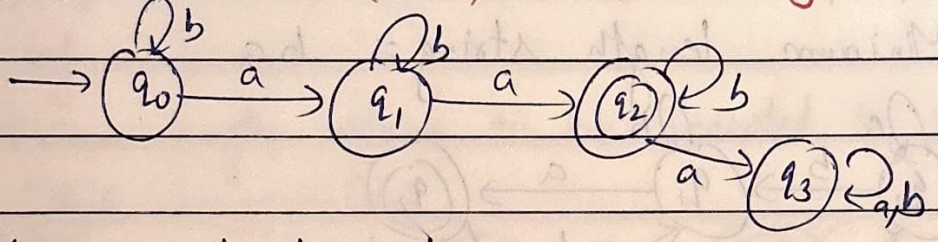


7) $L = \{w \mid w \in \Sigma^*, \Sigma = \{0,1\}$, w have exactly three consecutive 0's$\}$

Minimum length string : 000, 0001, 1000,



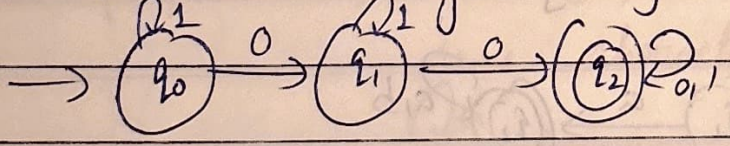
8) $L = \{w \mid w \in \Sigma^*, \Sigma = \{0,1\}$, w has exactly two zero's$\}$



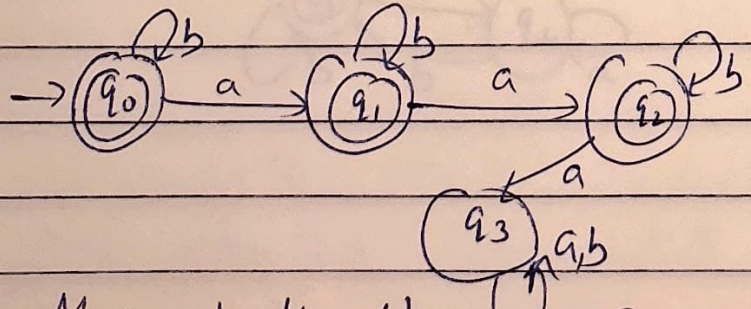
Minimum length string = aa

9) $L = \{w \mid w \in \Sigma^*, \Sigma = \{0,1\}$, w have atleast two zero's$\}$

Minimum length string = aa



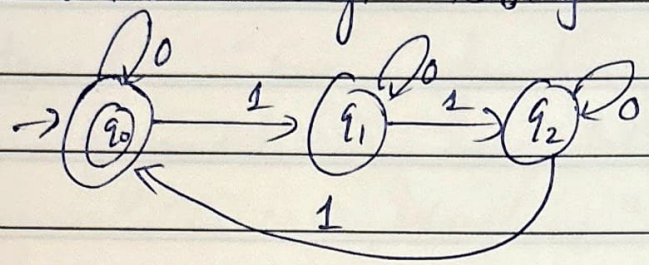
10) $L = \{w \mid w \in \Sigma^*, \Sigma = \{0,1\}$, w have atmost 2 zero's$\}$



Minimum length string = ϵ, a, b, bb ——

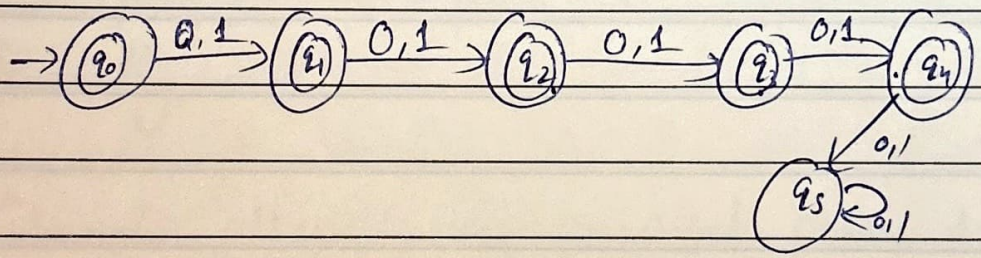
11.) $L = \{w / w \in \Sigma^*, \Sigma = \{0,1\}$, No. of 1's in w is divisible by 3.

Minimum length string = ϵ



12.) $L = \{w / w \in \Sigma^*, \Sigma = \{0,1\}, |w| \leq 4\}$

Minimum length string = ϵ



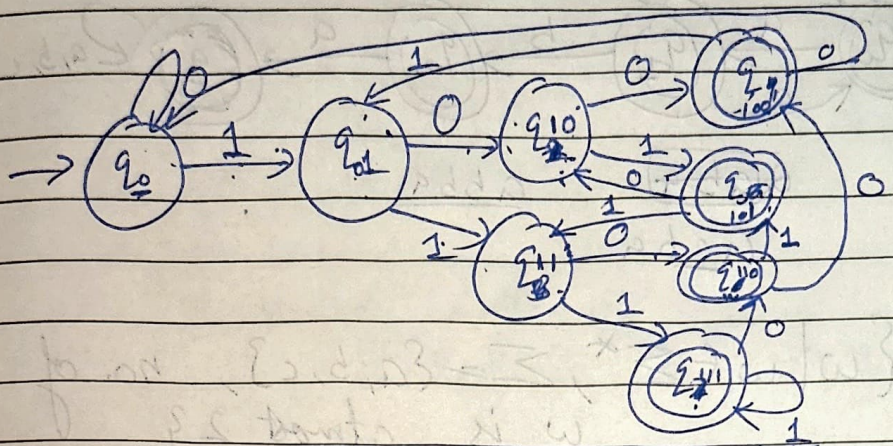
13.)

$1 \times 2 \times 2 \times 2 = 8$

$L = \{w \mid w \in \Sigma^*, \Sigma = \{0,1\}$, and third last digit is 1$\}$

$L = \{100, 101, 110, 111\}$

000 1 000 100



000
1001
1010
1011
1101
1100
1110

0 0 1 0 1 0 0 1 1 0 1 1 0 0 1 0 0

$q_0\ q_0\ q_0\ q_1\ q_{10}\ q_{01}\ q_{10}\ q_{01}\ q_{11}\ q_{10}\ q_{01}\ q_{11}\ q_{10}\ q_{01}\ q_{10}\ q_{01}\ q_{10}$

$L = \{w \mid w \in \Sigma^*, \Sigma = \{0,1\}$, 4th last symbol in w is 1$\}$

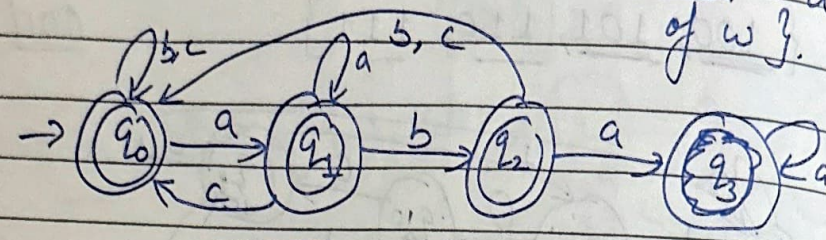
$L = \{1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$

	0	1
$\rightarrow q_0$	q_0	q_{01}
q_{01}	q_{10}	q_{11}
q_{11}	q_{110}	q_{111}
q_{10}	q_{100}	q_{101}
q_{110}	q_{1100}	q_{1101}
q_{111}	q_{1110}	q_{1111}
q_{100}	q_{1000}	q_{1001}
q_{101}	q_{1010}	q_{1011}
$*q_{1100}$	q_{1000}	q_{1001}
$*q_{1101}$	q_{1010}	q_{1011}
$*q_{1110}$	q_{1100}	q_{1101}
$*q_{1111}$	q_{1110}	q_{1111}
$*q_{1000}$	q_0	q_{01}
$*q_{1001}$	q_{10}	q_{11}
$*q_{1010}$	q_{100}	q_{101}
$*q_{1011}$	q_{110}	q_{111}

$\Sigma = \{0, 1, 2\}$

$1 \times 3 \times 3$

$L = \{w \mid w \in \Sigma^*, \Sigma = \{a, b, c\}$, 'aba' should not be a substring of $w\}$.



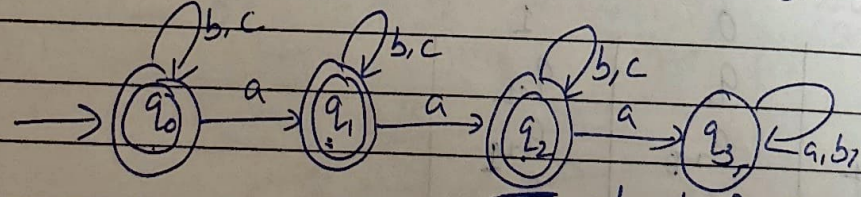
~~aaba~~ abba

acba

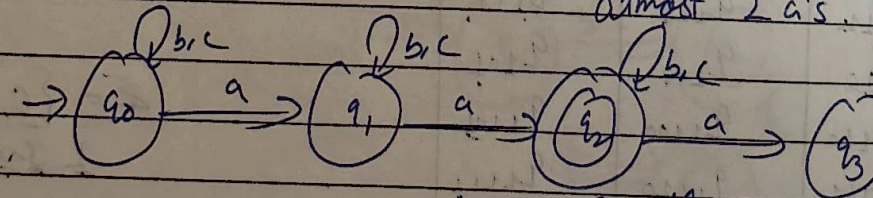
$L = \{w \mid w \in \Sigma^*, \Sigma = \{a, b, c\}$, no. of a's in w is atmost 2$\}$

$L = \{w \mid w \in \Sigma^*, \Sigma = \{a, b, c\}$, no. of a's in w is exactly 2$\}$.

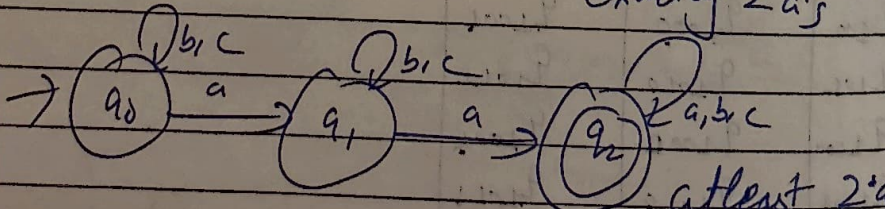
$L = \{w \mid w \in \Sigma^*, \Sigma = \{a, b, c\}$, no. of a's in w is atleast 2$\}$.



atmost 2 a's.

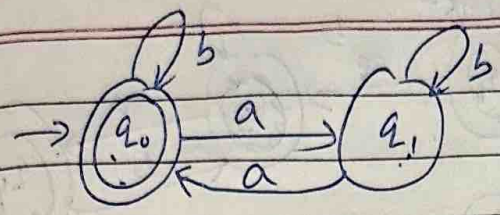


Exactly 2 a's



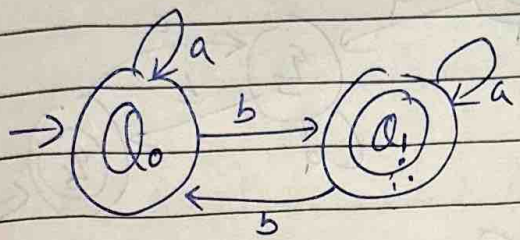
atleast 2 a's

$L = \{w \mid w \in \Sigma^*, \Sigma = \{a, b\}$, no. of a's in w is even & no. of b's in w is odd$\}$.



even no. of a's

AND / OR



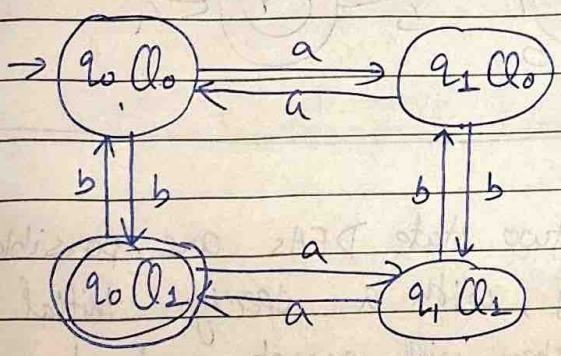
Odd no. of b's.

Compound Automata

$\{q_0, q_1\} \times \{q_0, q_1\}$

$\{(q_0,q_0), (q_0,q_1), (q_1,q_0), (q_1,q_1)\}$

$q_1\ q_0$



$L = \{ \overline{w} \mid w \in \Sigma^*, \Sigma = \{a,b,c\},$ ~~no. of a's in~~ this $|w_a| \bmod 2 = 1,$
$|w_b| \bmod 3 = 2,$
$|w_c| \bmod 3 = 0 \}$

$L = \{w \mid w \in \Sigma^*, \Sigma = \{a,b\},$ w starts & ends with same letter $\}$

$L = \{w \mid w \in \Sigma^*, \Sigma = \{a,b\},$ w starts & ends with different letter $\}$