Unit - 1: Introduction to Finite Automata.

Basic concepts to be discussed like Set, Subset, power set, difference of two sets, complement of set.

What is Automata?

- Abstract model of the digital computer

* Automata Theory is the study of abstract computing devices or abstract machine. (Theoretical concept)

Before there were computers, A. turing studied an abstract machine that had all capabilities of today's computers. Turing's goal was to describe what a computing machine could do and what it could not do, his conclusions apply not only to abstract machines but to today's real machines also. All these theoretical developments (concepts) bear directly on what computer scientists do today. Some concepts like finite automata & formal grammars are used in the design and construction of important kinds of software. Some concepts like turing machine help us understand what we can expect from our software.

Basic Notations in TAFL:

1) Alphabet: An Alphabet is defined as finite, non-empty set of symbols.
Σ denotes the alphabet.
Eg: $\Sigma = \{0, 1\}$ is the binary alphabet
$\Sigma = \{a, b, \ldots z\}$ is the set of all lower-case letters.
$\Sigma = \{a, b \ldots z, A, B \ldots Z, 0, 1, \ldots 9, C,), \ldots, <, > \ldots\}$ → set of alphabets of C language.

2) String: A string (sometimes word) is defined as finite

Sequence of symbols chosen from the alphabet Σ.

Eg: 010 is the string from the binary alphabet $\Sigma = \{0,1\}$

$\Sigma = \{0,1\}$, the strings obtained from Σ are ↓

$\{0, 1, 00, 01, 10, 11, 010101, \ldots\}$ i.e $w = 010$

Empty string is the string with zero occurances of symbols. Denoted by the symbol ϵ

Note: → Lower case letters a, b, c... along with the symbols such as +, −, (,), [, ... are used to denote the symbols in Σ

→ Lower case letters such as w, x, y, z are used to indicate the strings.

3) Concatenation of strings:

Concatenation of two strings x and y denoted by xy is the string obtained by writing the (letters) symbols of string x followed by the symbols of string y.

— i.e String formed by making a copy of x and following it by a copy of string y.

Eg: $x = a_1 a_2 \ldots a_n$ $\quad x = 1101$

$y = b_1 b_2 \ldots b_n$ $\quad y = 0011$

$xy = a_1 a_2 \ldots a_n b_1 b_2 \ldots b_n$ $\quad xy = 11010011$

4) Substring, prefix and suffix

Let w be the string obtained from the symbols in Σ. The string w if it can be decomposed into three strings x, y and z such that

$w = xyz$

then x is a substring, y is a substring

A prefix is string of any number of leading symbols

Eg: $w = xyz$.

The prefixes are ϵ, x, xy, xyz

Suffix is a string of any number of trailing symbols.

Eg: $w = xyz$

Suffixes are ϵ, z, yz, xyz

5) Reverse of a string is the string obtained by writing the symbols in reverse order.

Eg: $w = xyz$

$w^R = zyx$

6) Length of a string defined as the number of positions for symbols in the string.

Eg: $w = 01101$ is the string with the length 5

- Standard notation for the length of a string w is $|w|$

7) Power of an Alphabet: If Σ_1 is an alphabet, The power of an alphabet denoted by Σ_1^i is the set of all strings of length i from that alphabet.

Eg: If $\Sigma_1 = \{0,1\}$ then

$\Sigma_1^0 = \{\epsilon\}$ denotes set of strings of length 0

$\Sigma_1^1 = \{0,1\}$

$\Sigma_1^2 = \{00,01,10,11\}$

2 Variations of power of an alphabet:

- Kleene Closure (Kleene star/star operator)
- Kleene Plus

Kleene Closure is defined as

$$\Sigma_1^* = \Sigma_1^0 \cup \Sigma_1^1 \cup \Sigma_1^2 \cup \Sigma_1^3 \cup \ldots$$ (of any length)

which is the set of all strings of any length including the null string over an alphabet Σ_1

Eg: Σ_1^* for $\Sigma_1 = \{0,1\}$ is

$\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$

Kleene Plus is defined as

$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$$

which is the set of all strings excluding the empty string from the set of strings.

Note: $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$

8) Language : A set of strings all of which are chosen (obtained) from Σ^*, where Σ is a particular alphabet is called a language.

- If Σ is an alphabet and $L \subseteq \Sigma^*$ then L is a language over Σ. A language over Σ need not include strings with all the symbols of Σ.

Eg: 1) The language of all strings consisting of n 0's followed by n 1's for some $n \geq 0$

$\{\epsilon, 01, 0011, 000111, \dots\}$

2) The set of strings of 0's and 1's with an equal number of each is $L = \{\epsilon, 01, 10, 0011, 0101, \dots\}$

3) ϕ, the empty language, is a language over any alphabet.

4) $\{\epsilon\}$, the language consisting of only the empty string

Note: $\phi \neq \{\epsilon\}$

Types of Languages

- Type 0 Language (Phrase structured language)
- Type 1 Language (Context Sensitive Language)
- Type 2 Language (Context Free Language)
- Type 3 Language (Regular Language)

Regular Languages: The regular languages are defined as the languages accepted by DFA's, NFA's and ϵ-NFA's and also the languages defined by regular expressions.

- Regular languages can be described using four different methods:

Finite Automata
- DFA (Deterministic Finite Automata)
- NFA (Non-deterministic Finite Automata)
- ϵ-NFA (Non-deterministic Finite Automata with ϵ-moves)
- Regular Expressions

Abstract Machine:

Abstract is nothing but the theoretical concept. Abstract Machine (Abstract Computer) is a conceptual or theoretical model of a computer hardware or software system which really does not exist. These machines are not actual machines but have some features which are found in real machines. The various types of abstract machines (abstract computing devices) are:

- Finite Automata
- Linear Bounded Automata
- Push Down Automata
- Turing Machine

Finite Automata: A finite automata is the abstract / mathematical model which is used to study the abstract machine (computing device) with the inputs chosen from Σ which involves states and transitions

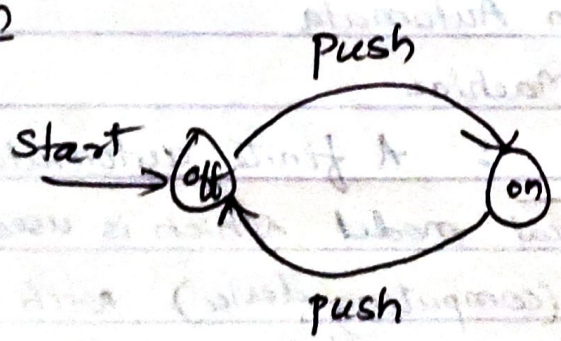
among states in response to the inputs.

Why finite Automata? Where it is used?

Finite Automata are useful model for design & construction of many important kinds of Hardware and Software. Some of the important kinds:

1) Software for designing and checking the behaviour of digital circuits.
2) The lexical analyser of the compiler, that is the compiler that breaks input text into logical units like keywords, identifiers etc.
3) Software for checking and verifying the systems of all types that have a finite number of states such as communication protocols or protocols for secure exchange of information [Software for checking the correctness of correctness of communication protocols]
4) Software for scanning large bodies of text, such as collections of web pages to find the occurrences of words, phrases etc...

→ Simplest example for finite Automata is: an on/off Switch



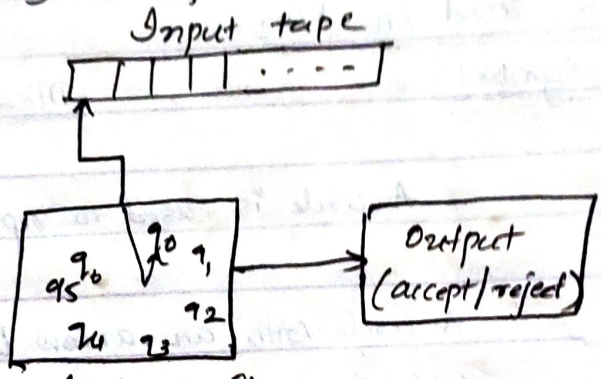
→ Circles represent states
→ labels associated with the arcs between the states represents the input given to this go to another state.

Depending upon the state of the switch, that is if the switch is in the off state, pressing the button changes it to the on state, and if the switch is in the on state, pressing the same button turns it to the off state.

Finite Automata has 3 components:

1) Input file
2) Control unit
3) Output.

Block diagram of FA is



Input tape: The input tape is divided into cells each of which can hold one symbol.

Control Unit: The machine has some states one of which is the start state designated as q_0 and atleast one final state. Apart from these states, it has some finite states designated by $q_1, q_2 \ldots$ Based on the current i/p symbol, the state of the machine can change.

Output: Output may be accept or reject. When end of input (string) is encountered, the string (o/p) is accepted if & only if the automata will be in one of its final states. Otherwise, the string is rejected.

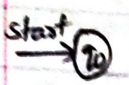
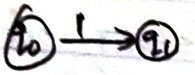
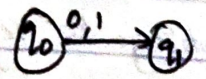
Working: The machine is assumed to be in start state q_0.

⇒ The i/p pointer points to the first symbol of the

tape pointing to the string to be processed.
After scanning the current i/p symbol, the machine can enter into any of the states $q_0, q_1, q_2 \ldots$ and the input pointer automatically points to the next character by moving one cell towards right.
When the end of the string is encountered, the string is accepted if & only if the automata is in one of the final states. Otherwise, the string is rejected.

For the simplest example given,

Symbols used in FA:

Symbol	Meaning
a) 	A circle is used to represent a state.
b) 	A circle with an arrow labeled with start which is not originating from any state. represents the start state of the machine.
c) 	Two circles are used to represent final state
d) 	An arc with label 1 goes from state q_0 to state q_1. This indicates that there is a transition from state q_0 on i/p symbol 1 to state q_1. Representation is $\delta(q_0, 1) = q_1$
e) 	An arc with label 0 indicates the machine in state q_0 on reading a 0, remains in state q_0
f) 	An arc with label 0,1 indicates that the machine in state q_0 on reading a 0 or 1 enters into state q_1. $\delta(q_0, 0) = q_1$ $\delta(q_0, 1) = q_1$

Deterministic Finite Automata (DFA)

Definition : Deterministic Finite Automata in short DFA is a 5-tuple or quintuple defined as

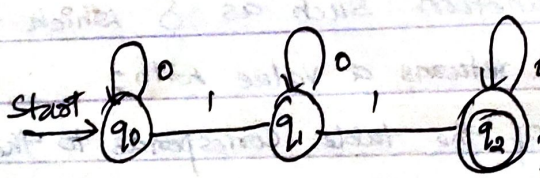
$$M = (Q, \Sigma, \delta, q_0, F)$$

where
- M is the name of the machine
- Q is the finite, non-empty set of states
- Σ is the finite, non-empty set of input symbols
- $\delta : Q \times \Sigma$ to Q is a transition function which is a mapping from $Q \times \Sigma$ to Q. Based on the current state and i/p symbol, the machine enters into another state
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accepting or final states.

The Name DFA emerges from :

Deterministic : For each input symbol there is exactly one transition from the current state. So the machine is deterministic. Since it is finite, it has finite number of states and transitions. Hence the name DFA.

Eg:



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

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

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

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

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

$\delta \quad Q \times \Sigma = 6$

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

Simple Notations for DFA:

DFA can be represented as:

- Transition diagram (Transition graph)
- Transition Table.

Transition Diagram:

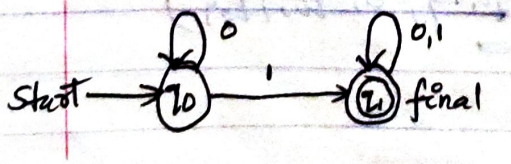
Transition diagram for DFA $M = (Q, \Sigma, \delta, q_0, F)$ is defined as graph with circles, arrows & arcs with labels, two circles. It is formally defined as:

- Each state of Q corresponds to node represented using circle.
- Alphabets in Σ are represented as labels along with arcs.
- The transition from one state to other is indicated by the directed arc.
- The start state is a state which has an arrow not originating from any node and is labeled with start.
- The final states are represented by double circles.

Transition Table:

Transition table for DFA $M = (Q, \Sigma, \delta, q_0, F)$ is defined as conventional, tabular representation of a transition function such as δ which takes two arguments and returns a value with:

- The rows of the table corresponds to the states of DFA
- The columns corresponds to the input symbols obtaind from Σ
- If q is the current state and a is the current input symbol, the value returned from $\delta(q, a) = p$ represents the next state "p" of (machine) DFA and is entered in row q and column a
- The state marked with an arrow is start state
- The state marked with star is final state/accepting state



Transition Diagram

δ	0	1
→ q_0	q_0	q_1
* q_1	q_1	q_1

Transition Table.

Extended Transition Function of DFA to Strings:

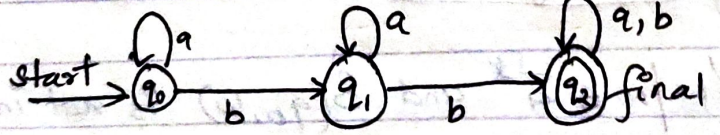
$\delta(q,a) = p$ is a transition function which accepts two parameters namely state q and input symbol a and returns a state p which is the next state of machine. But if there is a change of state from q to state p on input string w, then we use extended transition function.

Extended Transition Function $\hat{\delta}$ or δ^* describes what happens to a state of machine when the input is a string. The extended transition function $\delta^* : Q \times \Sigma^* \to Q$ is defined recursively as:

- **Basis:** $\boxed{\hat{\delta}(q,\epsilon) = q}$ This indicates that if the machine is in state q and reads no input, then the machine is still in state q
- **Induction:** Let $w = xa$ where a is the ~~input~~ last symbol of w and x is the remaining string of w. Then

$$\boxed{\hat{\delta}(q,w) = \hat{\delta}(q,xa) = \delta(\hat{\delta}(q,x),a) = p}$$

How the strings are processed by DFA ??



⇝ Process string abab using

a) Transition function:

$\delta(q_0, a) = q_0$

$\delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1$

$\delta(q_1, b) = q_2$

b) Extended Transition Function:

For prefix ϵ: $\hat{\delta}(q_0, \epsilon) = q_0$

For prefix a: $\hat{\delta}(q_0, a) = \delta(\hat{\delta}(q_0, \epsilon), a)$
$= \delta(q_0, a) = q_0$

For prefix ab: $\hat{\delta}(q_0, ab) = \delta(\hat{\delta}(q_0, a), b)$
$= \delta(q_0, b) = q_1$

For prefix aba: $\hat{\delta}(q_0, aba) = \delta(\hat{\delta}(q_0, ab), a)$
$= \delta(q_1, a) = q_1$

For prefix abab: $\hat{\delta}(q_0, abab) = \delta(\hat{\delta}(q_0, aba), b)$
$= \delta(q_1, b) = q_2$

Language accepted by DFA:

The language of the DFA $M = (Q, \Sigma, \delta, q_0, F)$ denoted by $L(M)$ is defined as

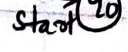
$$L(M) = \{ w \mid w \in \Sigma^* \text{ and } \hat{\delta}(q_0, w) \text{ is in } F \}$$

which is the set of strings which takes the machine from initial state to final state.

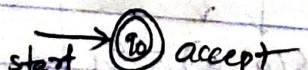
Language rejected by DFA

$$L(M) = \{ w \mid w \in \Sigma^* \text{ and } \hat{\delta}(q_0, w) \text{ is not in } F \}$$

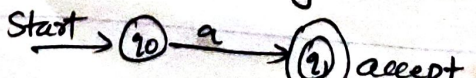
* DFA to accept empty string (language) $L = \{\phi\}$ is



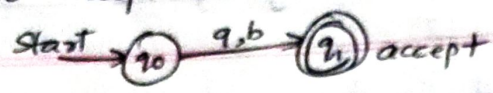
* DFA to accept an empty string $L = \{\epsilon\}$ is



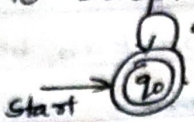
* DFA to accept exactly one a is



* DFA to accept one a or one b is



* DFA to accept zero or more a's is



DFA Design Techniques

The various types of problems for which we construct DFA is

- Pattern recognition problems
- Divisible by k problems
- Modulo - k - counter problems

Pattern Recognition Problems:

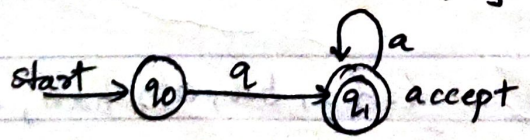
Steps involved:

1) Identify the minimum string
2) Identify the alphabet
3) Construct a Skeleton DFA
4) Identify other transitions not defined in step3
5) Construct DFA using transitions of step3 & step4

Q1) Construct an DFA to accept strings of a's having atleast one a

Minimum string is a

Alphabet is $\Sigma = \{a\}$



Write DFA as $M = (Q, \Sigma, \delta, q_0, F)$ for every Question

Strings accepted are a, aa, aaa, aaaa....

$L = \{a, aa, aaa, aaaa, \ldots\}$

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

$\Rightarrow L = \{a^n \mid n \geq 1\}$

$\Rightarrow L = \{w : n_a(w) \geq 1, w \in \{a\}^*\}$

(OR)

$L = \{w : \exists n \geq 1 \ (w = a^n)\}$

δ	a
$\rightarrow q_0$	q_1
$*q_1$	q_1

Q2) Construct DFA to accept strings of a's and b's having at least one a

Minimum string is a

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

Strings accepted by DFA are

a, ab, ba, bab, baa, aa, aba, aaba

$L = \{a, ab, ba, bab, baa, aa, aba, aaba...\}$

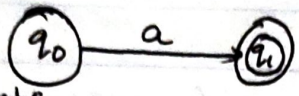
$\bar{L} = \{\epsilon, b, bb, ...\}$

$\delta(q_0, b) = ?$ cant reach State q_1 Since b is not accepted by DFA. Hence $\delta(q_0, b) = q_0$ itself

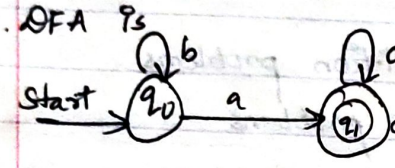
$\delta(q_1, a) = ?$ Cant reach state q_0 Since for string aa if it will reach state q_0 then its not in final State. But aa is string which is accepted by DFA hence it must reach final state q_1

$\delta(q_1, b) = q_1$ itself.

Skeleton DFA for minimum string:



∴ DFA is



δ	a	b
→ q_0	q_1	q_0
* q_1	q_1	q_1

$\therefore L = \{b^n a (a+b)^m \mid n, m \geq 0\}$

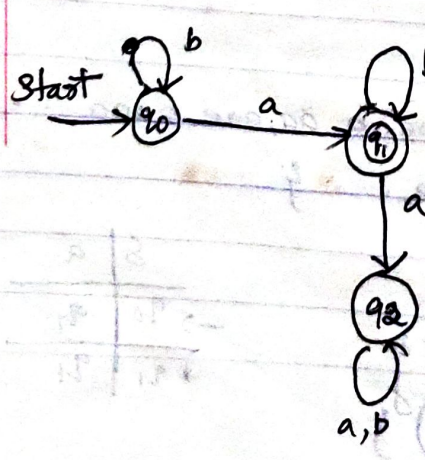
$L = \{w : n_a(w) \geq 1, w \in \{a, b\}^*\}$

Q3) Construct DFA to accept strings of a's and b's having exactly one a.

Minimum string a

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

$\delta(q_0, b) = q_0$ itself



$L = \{a, ba, ab, bab, bbab, ...\}$

$\bar{L} = \{\epsilon, bb, aa, baa ...\}$

$\delta(q_0, b) = q_1$

$\delta(q_1, a) = ?$ if cant reach q_0 since only single a is accepted.

$\therefore \delta(q_1, a) = q_2$

δ	a	b
→ q_0	q_1	q_0
* q_1	-	q_1

δ	a	b
→ q_0	q_1	q_0
* q_1	q_2	q_1
q_2	q_2	q_2

$\therefore L = \{w : n_a(w) = 1,\ w \in \{a,b\}^*\}$

$L = \{b^m a b^n \mid m, n \geq 0\}$

In the above DFA, State q_2 is called as trap state/ dead state.

- for any of the input symbols, there is no escape from this state.

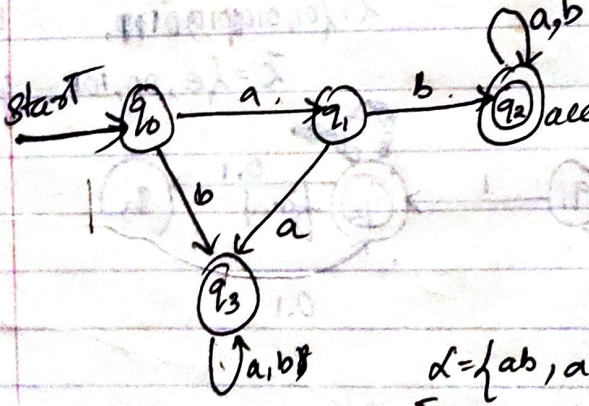
Trap state: State for which there exists transition to itself for all the input symbols chosen from Σ

- The non final trap state is called as dead state.

Q4) Obtain DFA to accept strings of a's & b's starting with the ab.

Minimum string is ab

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



$\delta(q_0, b) \neq q_0$ since it starts from ab string

$\therefore \delta(q_0, b) =$ - i.e q_3 (trap state)

$\delta(q_1, a) =$? it can't reach q_1 itself nor it can't reach q_0 also [since abab and is not accepted]

$L = \{ab, aba, abb, abab \ldots\}$

$L = \{\epsilon, ba, bab, \ldots\}$.

$L = \{ab(a+b)^n \mid n \geq 0\}$

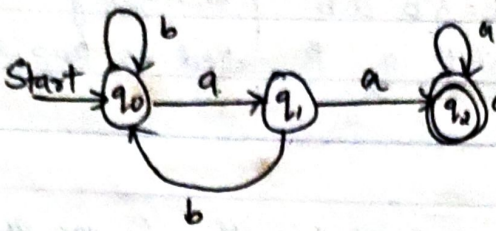
δ	a	b
→ q_0	q_1	q_3
* q_1	q_3	q_2
* q_2	q_2	q_2
q_3	q_3	q_3

Ending with ab

Q6) DFA to accept strings of a's and b's having substring aa.

Minimum string is aa

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



Strings accepted are

$L = \{aa, baa, baab, aab, aaa\}$

$L = \{\epsilon, ab, bab, aba, \ldots\}$

$\delta(q_0, b) = ?$ Can reach to q_1 since abaa string will be accepted.

δ	a	b
→ q_0	q_1	q_0
q_1	q_2	q_0
* q_2	q_2	q_2

$L = \{ (a+b)^m aa (a+b)^n \mid m, n \geq 0 \}$

Q7) DFA to accept strings of even length begining with 01

SP

Given: $L = \{w \mid w$ is of even length and begins with 01$\}$

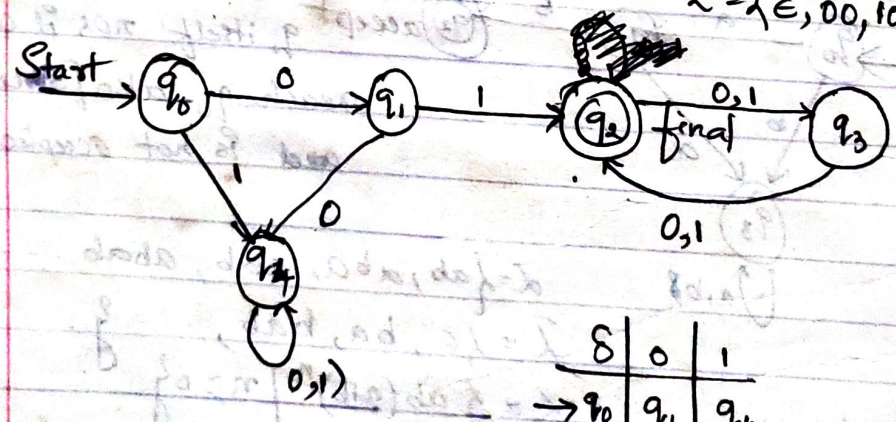
Minimum string is 01

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

Strings accepted are

$L = \{01, 0100, 0101, 0110, \ldots\}$

$L = \{\epsilon, 00, 101 \ldots\}$

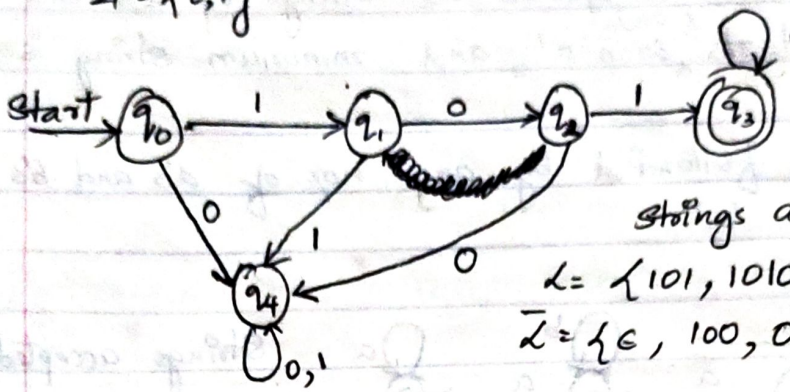


δ	0	1
→ q_0	q_1	q_4
q_1	q_2	q_4
* q_2	q_3	q_3
q_3	q_2	q_2
q_4	q_4	q_4

Q8) DFA to accept strings of 0's & 1's begining with 101

Minimum string is 101

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



Strings accepted are

$L = \{101, 1010, 1011, 10100 \ldots\}$

$\bar{L} = \{\epsilon, 100, 010, \ldots\}$.

$L = \{101(0+1)^n \mid n \geq 0\}$

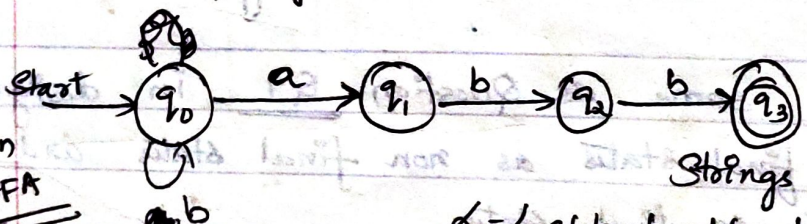
δ	0	1
$\rightarrow q_0$	q_4	q_1
q_1	q_2	q_4
q_2	q_4	q_3
$*\ q_3$	q_3	q_3
q_4	q_4	q_4

Q9) DFA to accept strings of a's & b's ending with the string abb

Minimum string is abb

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

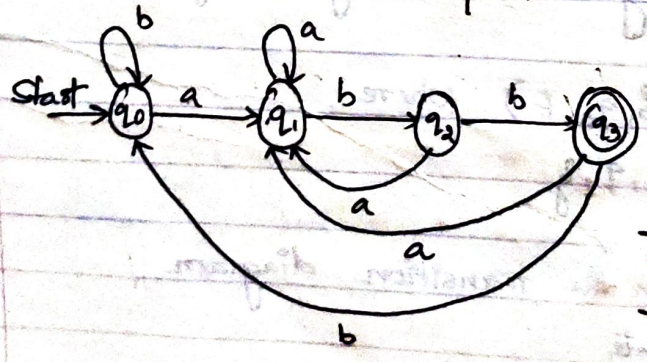
Skeleton DFA



Strings accepted are

$L = \{abb, babb, baabb, baaaabb$

$\bar{L} = \{\epsilon, baa, babba \ldots\}$



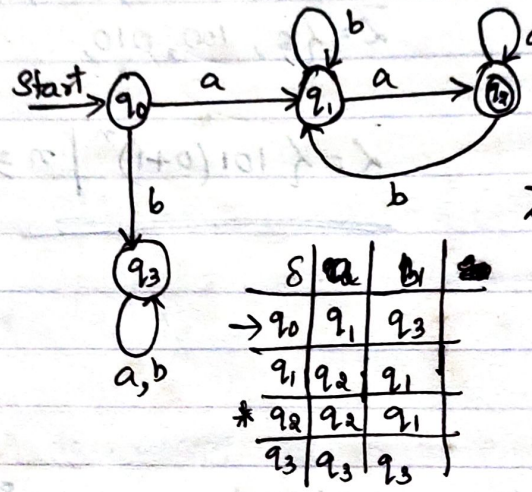
δ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
$*\ q_3$	q_1	q_0

$L = \{(a+b)^n abb \mid n \geq 0\}$

Q)10) Draw a DFA to accept strings of a's & b's such that $L = \{awa \mid w \in (a+b)^n$ where $n \geq 0\}$

Soln: Language is interpreted as strings of a's and b's which starts & ends with 'a' and minimum string is aa

Begins with 'a' is followed by any no. of a's and b's and ends with 'a'.



δ	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_2	q_1
$*\ q_2$	q_2	q_1
q_3	q_3	q_3

Strings accepted are

$L = \{aa, aba, aaba, abbaba, \ldots\}$

$\bar{L} = \{\epsilon, ab, ba, abab, \ldots\}$

Q)11) Draw a DFA to accept strings of a's and b's which do not end with the string abb

Soln: DFA is same as Question Q9 in design But make the final states as non final states and non-final states as final states.

The resulting DFA is given by

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

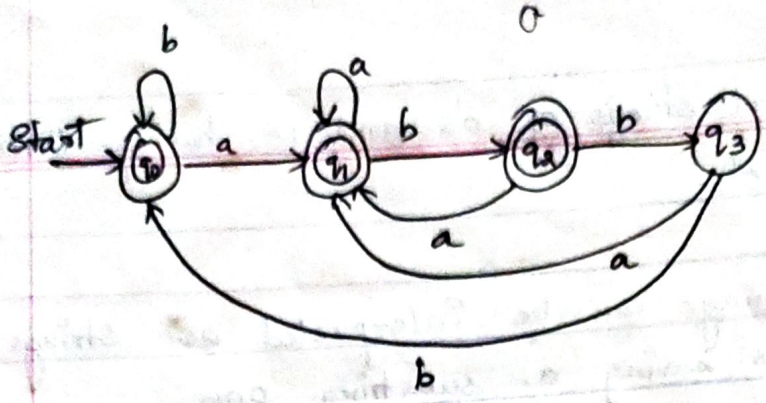
$Q = \{q_0, q_1, q_2, q_3\}$

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

δ is shown in the transition diagram

q_0 is start state

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



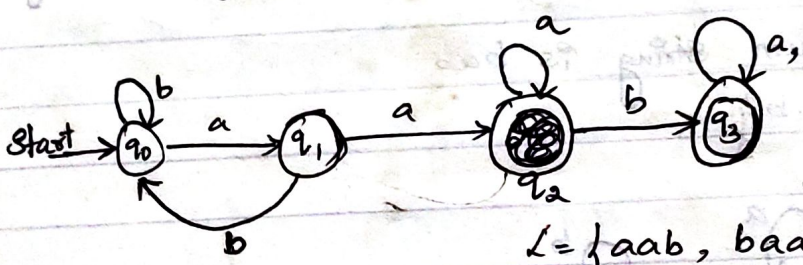
δ	a	b
→ * q_0	q_1	q_0
* q_1	q_1	q_2
* q_2	q_1	q_3
q_3	q_1	q_0

H.W

Q)12) Draw a DFA to accept strings of a's and b's having a substring aab

Soln: Minimum string accepted is aab

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



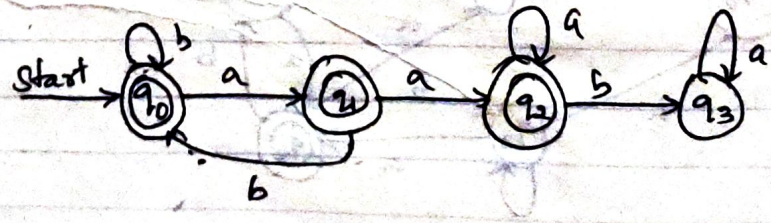
Strings accepted

$L = \{aab, baaba, aaabab \ldots\}$

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

H.W

Q13) DFA to accept strings of a's & b's except those having the substring aab.



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