1.1 BASIC CONCEPTS AND FINITE AUTOMATA

Basic Concepts

Alphabet (Σ)

It is defined as the finite set of i/p symbols

e.g. : $\Sigma = \{0, 1, 2\}$

String / sentence / word

It is defined as the finite sequence of symbols over the given Σ

e.g. : 0, 1, 2, 00, 111, 201, 220, ...

String length

It is defined as the number of symbols present in given string.

e.g. x = 12012

$|x| = 5$

Note : The string of length zero (0) would be denoted by $\in$ (Epsilon)

Language

It is defined as the set of strings defined over the given Σ (alphabet)

$$\text{e.g. : } L = \left\{ x \;\middle|\; \begin{array}{c} x \text{ ends in "ba"} \\ \text{over } \Sigma = \{a, b\} \end{array} \right\}$$

$$L = \{ba, aba, bba, aaba, \ldots.\}$$

Operation on languages

(1) Union of 2 languages

$$L_1 \cup L_2 = \left\{ x, y \;\middle|\; \begin{array}{c} x \in L_1 \\ \& \\ y \in L_2 \end{array} \right\}$$

(2) Concatenation of 2 languages

$$L_1 L_2 = L_1 L_2 = \left\{ xy \;\middle|\; \begin{array}{c} x \in L_1 \\ \& \\ y \in L_2 \end{array} \right\}$$

Example on union and concatenation of 2 languages

e.g. : $L_1 = \{00, 10, 110\}$

$L_2 = \{aa, ba\}$

$L_1 \cup L_2 = \{00, 10, 110, aa, ba\}$

$L_1 L_2 = \{00aa, 00ba, 10aa, 10ba, 110aa, 110ba\}$

(3) Closure of a language

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

(4) Positive closure of a language

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

Examples

$L = \{ba\} = L^1$

$L^3 = L.L.L. = \{bababa\}$

$L^2 = L.L. = \{baba\}$

$L^0 = \{\in\}$

$L^* = L^0 \cup L^1 \cup L^2 \cup \ldots\ldots$

$= \{\in, ba, baba, \ldots..\}$

$L^+ = L^1 \cup L^2 \cup L^3 \cup \ldots.$

$= \{ba, baba, bababa, \ldots\ldots\}$

1.1.1 Finite Automata (FA)

Finite Automata is considered to be a mathematical model of machine or a system.

Model of (FA)

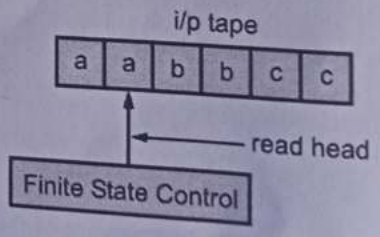


Fig. 1.1.1 : Model of FA

Component of FA

F.A. consists of finite set of states, i/p tape and a read head.

Working of F.A.

Depending on the state and i/p symbol :

F.A. can change the state or remain in same state.

F.A. moves the head to the right of current cell by 1.

▶ **Variations of F.A.**

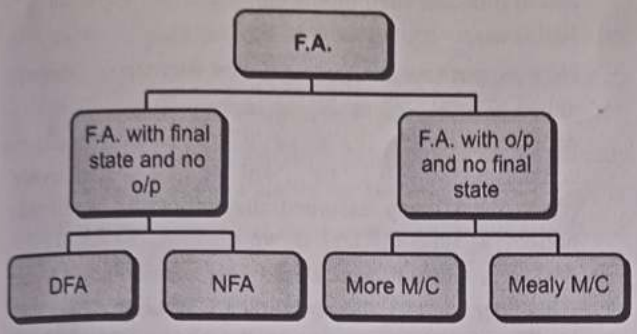


Fig. 1.1.2

1.1.2 Finite State Machine

❒ **Definition :** FSM consists of finite set of state(S) that alter on receiving the I/p set (I) to produce the output set (O)

FSM defines two functions :-

1. **State function :** STF : $S \times I \rightarrow S$
2. **Machine function :** MAF : $S \times I \rightarrow O$

1.1.3 Solved Examples in FSM

Ex. 1.1.1 : Design FSM to check whether the given decimal number is divisible by 3.

☑ **Soln. :**

▶ **Step 1 :** Theory (Definition of FSM)

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

▶ **Step 2 :** Logic

Input (I) = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Note : In the given sum input is decimal number, therefore here input is from 0 to 9.

Output (O) = {Y, N}

State (S) = $\{q_s, q_0, q_1, q_2\}$

(q_s is the start stage q_0 is the state where remainder is 0. q_1 is the state where remainder is 1. q_2 is the state where remainder is 2).

▶ **Step 3 :** Implementation

Note : In this sum, we need to check whether given decimal number is divisible by 3 or not and a number is said to be divisible by 3, when it is divided by 3 and we get zero (0) as remainder. And therefore here we are going to make use of mod function as mod function gives remainder as output when applied.

	S/I	{0 ,3, 6, 9}	{1, 4, 7}	{2, 5, 8}
R	$\rightarrow q_s$	q_0	q_1	q_2
0	q_0^*	q_0	q_1	q_2
1	q_1	q_1	q_2	q_0
2	q_2	q_2	q_0	q_1

Note : Here q_s represents start, start and there is no input already present on start state therefore if a number comes as input to start. state it is considered as it is.

- **For Example :** Suppose 3 comes as input to start state q_s and we perform mod function on it.
- i.e. 3%3 (% represents mod function) and we get remainder zero (0) as output and remainder 0 is assigned to state q_0. Therefore in the above table, one can see that state q_s over input {0, 3, 6, 9} gives state q_0 as output.
- Because either we perform 3%3 or 6%3 or 9%3 or 0%3 we get same output i.e. we get remainder 0. Therefore in the above table, state q_s over input {0, 3, 6, 9} gives state q_0 as output.
- Similarly when we are at state q_s and we get input either 1 or 4 or 7 and we perform 1%3 or 4%3 or 7%3, we get same output i.e. we get remainder 1 and remainder 1 is assigned to state q_1. Therefore in the above table, state q_s over input {1, 4, 7} gives state q_1 as output.
- Similarly when we are at state q_s and we get either input 2 or 5 or 8 and we perform 2%3 or 5%3 or 8%3, we get same output i.e. we get remainder 2 and remainder 2 is assigned to state q_2. Therefore in the above table state q_s over input {2, 5, 8} gives state q_2 as output.

- Now let us suppose we are at state q_0 and to reach state q_0 from start state q_s in this sum, is only possible if start state would have received an input which when divided by 3, gives us remainder 0 as output. In this sum such inputs can either be 0, 3, 6 or 9.
- For this instance, let us assume that input is 3. Therefore when we reach state q_0 from start state q_s, machine has already processed input 3. Therefore we already have input 3 present at state q_0 even before we receive any input at state q_0.
- Now let us assume we are at state q_0 and we receive 0 as our next input. Now since q_0 is not a start state, input '0' won't be considered as just 0, rather state q_0 has already input 3 present in this case.
- Therefore the number that would be considered will be '30' instead of '0' and now let us perform 30 %3, we would get remainder 0 as output and remainder 0 is assigned to state q_0, Therefore in the above table one can see state q_0 over input {0, 3, 6, 9} gives q_0 as output. Because instead of input 0 if our inputs would have been either 3, 6 or 9 they would have been considered as '33', '36' and '39' instead of '3', '6' or '9' respectively and when we perform 33 %3, 36 %3 or 39 %3 we would have got remainder 0 as output and since remainder 0 is assigned to state q_0. Therefore in the above table one can see state q_0 over input {0, 3, 6, 9} gives state q_0 as output
- Similarly now let us assume we are at state q_0 and we either receive '1', '4' or '7' as inputs. They would be considered as '31', '34' and '37' respectively. (Since we already have assumed that input 3 is already present at state q_0) and if we perform 31 % 3, 34 % 3 or 37 % 3 we would get same output i.e. remainder 1 and since remainder 1 is assigned to state q_1. Therefore in the above table one can see that state q_0 over input {1, 4, 7} gives state q_1 as output.
- Similarly now let us assume we are state q_0 and we receive either '2', '5' or '8' as inputs. They would be considered as '32', '35' or '38' respectively (since we already have assumed that input 3 is already present at state q_0) and if we perform 32 %3, 35 %3 or 38 %3, we would get same output i.e. remainder 2 and since remainder 2 is assigned to state q_2. Therefore one can see in the above table that state q_0 over input {2, 5, 8} gives q_2 as output.
- Now similarly for state q_1, we can assume input 1 is already present at state q_1 (since for reaching state q_1 from start state q_s, machine would have received an input which when divided by 3 gives remainder as 1 and in this sum such inputs either can be '1', '4' or '7'. In this case we assumed it to be 1),
- Now suppose we are at state q_1 and we receive numbers either '0', '3', '6' or '9' as inputs. They will not be considered as '0', '3', '6' or '9', rather they would be considered as '10', '13', '16' or '19' respectively. (since we already assumed that input '1' is already present at state q_1) and if we perform 10%3, 13%3, 16%3 or 19%3 we would get same output i.e. remainder 1 and since remainder 1 is assigned to state q_1. Therefore one can see in the above table state q_1 over input {0, 3, 6, 9} gives state q_1 as output.
- Similarly now we are at state q_1 and we would receive either '1', '4' or '7' as inputs. They will be considered as '11', '14' or '17' instead of '1', '4' or '7' (since we already have assumed that input '1' is already present at state q_1) and if we perform 11%3, 14%3 or 17%3, we would get same output i.e. remainder 2 and since remainder 2 is assigned to state q_2. Therefore one can see in the above table state q_1 over input {1, 4, 7} gives state q_2 as output.
- Now similarly we are at state q_1 and we receive either '2', '5' or '8' as inputs, they would be considered as '12', '15' or '18' instead of '2', '5' or '8' (since we have assumed that input '1' is already present at state q_1) and if we perform 12%3, 15%3 or 18%3, we would get same output i.e. remainder 0 and since remainder 0 is assigned to state q_0. Therefore one can see in the above table state q_1 over input {2, 5, 8} gives state q_0 as output.
- Now similarly for state q_2 we can assume input '2' is already present at state q_2 (since for reaching state q_2 from start state q_s, machine would have received an input which when divided by 3 gives remainder as 2 and in this sum such inputs either can be '2', '5' or '8'. In this case we assumed it to be 2).
- Now suppose we are at state q_2 and we receive either '0', '3', '6' or '9' as inputs. They would be considered as '20' '23', '26' or '29' respectively instead of '0', '3', '6' or '9' (since we have assumed input 2 is already present at state q_2) and if we perform 20%3, 23%3,

26%3 or 29%3, we would get same output i.e. remainder 2 and since remainder 2 is assigned to state q_2. Therefore one can see in the above table state q_2 over input {2, 5, 8} gives state q_2 as output

- Now suppose we are at state q_2 and we receive either '1','4' or '7' as inputs. They would be considered as '21', '24' or '27' respectively instead of '1', '4' or '7'. (Since we have assumed that input '2' is already present at state q_2) and if we perform 21%3, 24%3, or 27%3, we would get same output i.e. remainder 0 and since remainder 0 is assigned to state q_0. Therefore one can see in the above table state q_2 over input {1, 4, 7} gives state q_0 as output.
- Now suppose we are at state q_2 and we receive either '2', '5' or '8' as inputs. They would rather be considered as '22', '25' or '28' respectively instead of '2', '5' or '8'. (since we have already assumed that input '2' is already present at state q_2) and if we perform 22%3, 25%3 or 28%3, we would get same output i.e remainder 1 and since remainder '1' is assigned to state q_1. Therefore one can see in the above table state q_2 over input {2, 5, 8} gives state q_1 as output.

STF : S X I → S

S\I	{0, 3, 6, 9}	{1, 4, 7}	{2, 5, 8}
→ q_s	Y	N	N
q_0^*	Y	N	N
q_1	N	N	Y
q_2	N	Y	N

MAF = S X I → 0

Note : In Machine Function (MAF), wherever there is a final state (in this sum final state is q_0) there we would assign 'y' as output elsewhere 'N' as output.

Note : In the table, start state is denoted by '→' sign and in the transition diagram, start state is denoted by 'start' sign.

AND

In the table final states are denoted by asterisk i.e. '*' sign and in the transition diagram, final states are denoted by concentric circles i.e. by '◎' sign.

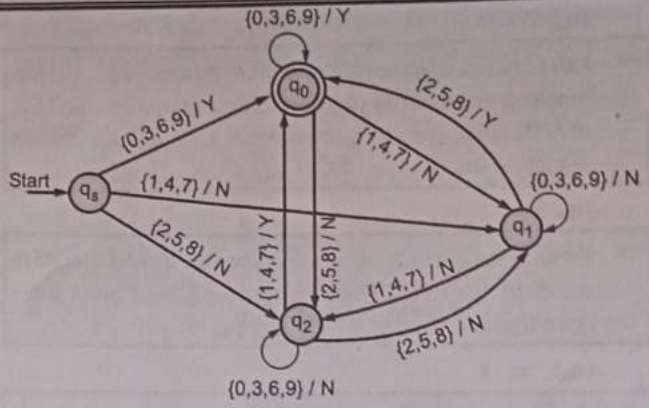
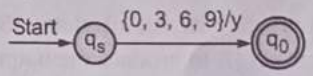


Fig. Ex. 1.1.1

Note : The transition diagram is drawn by replicating the STF and MAF tables for instance, if you look in STF table at state q_s and input {0, 3, 6, 9}

- state q_s over input {0, 3, 6, 9} gives q_0 as output and in MAF table state q_s over input {0, 3, 6, 9} gives 'y' as output. Therefore in the transition diagram it is represented as



(1A50A)Fig. Ex. 1.1.1(a)

Note : Since q_s is start state it is represented by 'start' → sign and q_0 is final state it is represented by '◎'.

In similar fashion we have completed the entire diagram.

▶ **Step 4 :** Example

$(q_S, 2169)$

Note : Here in example we have assumed input to be '2169' but FSM can process one input at a time. That means instead of processing entire input '2169' in one go, machine will first process input '2' then input '1' and so on.

Therefore here, FSM will first process $(q_s, 2)$. So when we look the state function table (STF table) at state q_s over input 2 we get state q_2 as output. Therefore $(q_s, 2169) \rightarrow (q_2, 169)$

$(q_2, 169)$

Note : Similarly now instead of processing $(q_2, 169)$ machine will process $(q_2, 1)$ and therefore when we look at STF table at state q_2 over input 1 we get state q_0 as output. Therefore, $(q_2, 169) \rightarrow (q_0, 69)$

$(q_0, 69)$

Note : Similarly now instead of processing $(q_0, 69)$ machine will process $(q_0, 6)$ and therefore when we look at STF table at state q_0 over input 6 we get state q_0 as output. Therefore, $(q_0, 69) \rightarrow (q_0, 9)$

$(q_0, 9)$

Note : Finally machine will process $(q_0, 9)$ and therefore when we look at STF table at state q_0 over input 9 we get state q_0 as output. Therefore, $(q_0, 9) \rightarrow (q_0)$

$(q_0) \Rightarrow Y$

Note : Since after processing the complete input machine is in the final state therefore machine is showing 'y' as output indicating yes, decimal number 2169 is divisible by 3.

Ex. 1.1.2 : Design FSM to check whether the given decimal number is divisible by 4.

☑ **Soln. :**

▶ **Step 1 :** Theory of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

▶ **Step 2 :** Logic $S = \{ q_s, q_0, q_1, q_2, q_3\}$

$I = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$O = \{Y, N\}$

▶ **Step 3 :** Implementation

	S\I	{0, 4, 8}	{1, 5, 9}	{2, 6}	{3, 7}
R	→ q_s	q_0	q_1	q_2	q_3
0	q_0^*	q_0	q_1	q_2	q_3
1	q_1	q_2	q_3	q_0	q_1
2	q_2	q_0	q_1	q_2	q_3
3	q_3	q_2	q_3	q_0	q_1

SFI = S X I → S

S\I	{0, 4, 8}	{1, 5, 9}	{2, 6}	{3, 7}
→ q_s	Y	N	N	N
q_0^*	Y	N	N	N
q_1	N	N	Y	N
q_2	Y	N	N	N
q_3	N	N	Y	N

MAF = S X I → 0

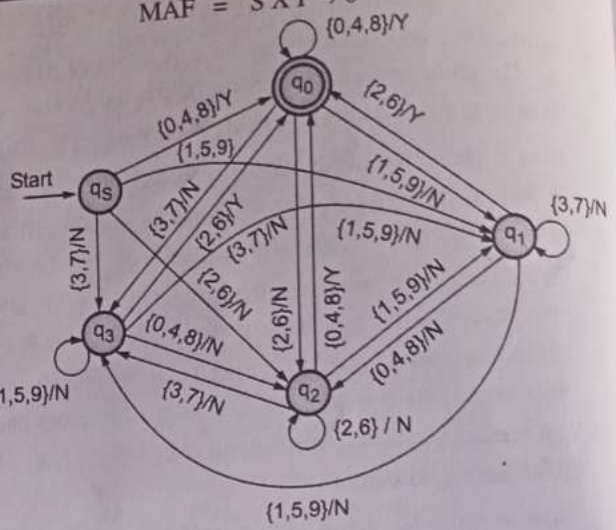


Fig. Ex. 1.1.2

▶ **Step 4 :** eg ;- $\{q_s, 57246\}$

$(q_1, 7246)$; $(q_1, 246)$

$(q_0, 46)$; $(q_0, 6)$

$(q_2) \rightarrow N$

Ex. 1.1.3 : Design a divisibility by 4 tester FSM for binary numbers.

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

▶ **Step 2 :** Logic

$I = (0, 1)$

Here the input is binary number, therefore input is either 0 or 1.

$O = \{Y, N\}$

$S = \{q_s, q_0, q_1, q_2, q_3\}$

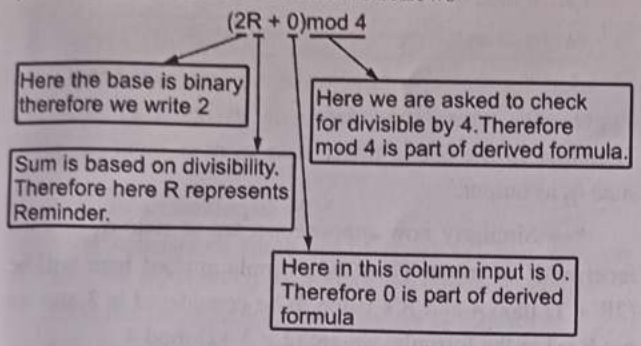
▶ **Step 3 :** Implementation

(1) STF : SXI → S

In sums of divisibility, apart from decimal number system for any number system we form a dynamic formula

to solve the sum. In this sum, the number system is binary therefore here also we will form a dynamic formula (dynamic means formula will change from sum to sum) to solve the sum.

For the column where input is 0 the formula applied is **(2R + 0) mod 4** and it is divided as follows



(1A51)**Fig. Ex. 1.1.3**

Similarly for the column where input is 1. The formula which will be applied is as follows :

(2R + 1) mod 4

		(2R + 0) mod 4	(2R + 1) mod 4
	S/I	0	1
R	$\rightarrow q_s$	q_0	q_1
0	q_0^*	q_0	q_1
1	q_1	q_2	q_3
2	q_2	q_0	q_1
3	q_3	q_2	q_3

The derived formula is not applicable for start state however since q_s is start state. So the inputs will be considered as it is and computed as follows :

- When state q_s will receive input 0 it will be considered as 0 and then we will perform 0%4, the output will be remainder 0 and since remainder 0 is assigned to state q_0. Therefore in the above table, state q_s over input 0 gives state q_0 as output.
- When state q_s will receive input 1 it will be considered as 1 and then we will perform 1%4, then output we receive will be remainder 1 and since remainder 1 is assigned to state q_1. Therefore in the above table, state q_s over input 1 gives state q_1 as output.

Note : For the remaining states we are going to apply the dynamic formula in following way.

- Since remainder 0 is assigned to state q_0. Therefore while applying the formula to state q_0, R's value in the formula will be considered as 0.

  * → Now suppose we are at state q0 and we receive input '0'. Then the formula applied here will be (2R + 0) mod 4 and here R's value to be considered is 0 and we put R = 0 in the formula, we get $(2 \times 0 + 0)$ mod 4 i.e. (0 + 0) mod 4

  i.e. 0% 4

  For operation 0 % 4, we get remainder 0 as output and since remainder 0 is assigned to state q_0. Therefore in the above table, state q_0 over input 0 gives state q_0 as output.

  * → Now suppose we are at state q_0 and we receive input '1'. Then the formula applied here will be (2R + 1) mod 4 and here R's value to be considered is 0 and we put R = 0 in the formula, we get

  $(2 \times 0 + 1)$ mod 4

  i.e. (0 + 1) mod 4

  i.e. 1 mod 4

  i.e. 1% 4

  And for operation 1%4, we get remainder 1 as output and since remainder 1 is assigned to state q_1. Therefore in the above table, state q_0 over input '1' gives state q_1 as output.

- Since remainder 1 is assigned to state q_1. Therefore while applying the formula to state q_1, R's value in the formula will be considered as 1.

  *→ Now suppose we are at state q_1 and we receive '0' as input. Then the formula applied here will be (2R + 0) mod 4 and here R's value to be considered is 1 and we put R =1 in the formula and we get

  $(2 \times 1 + 0)$ mod 4

  i.e. (2 + 0) mod 4

  i.e. 2 mod 4

  i.e. 2 % 4

  And for the operation 2%4, we get remainder 2 as output and since remainder 2 is assigned to state q_2. Therefore in the above table, state q_1 over input '0' gives state q_2 as output.

*→ Similarly now suppose we are at state q_1 and we receive '1' as input. Then the formula applied here will be (2R +1) mod 4 and here R's value to be considered is 1 and we put R = 1 in the formula, we get

$(2 \times 1 + 1) \bmod 4$

i.e. $(2 + 1) \bmod 4$

i.e. $3 \bmod 4$

i.e. $3\,\%\,4$

And for the operation 3%4, we get remainder 3 as output and since remainder 3 is assigned to state q_3. Therefore in the above table, state q_1 over input 1 gives state q_3 as output.

- Since remainder 2 is assigned to state q_2. Therefore while applying the formula to state q_2, R's value in the formula will be considered as 2.

*→ Now suppose we are at state q_2 and we receive '0' as input. Then the formula applied here will be (2R + 0) mod 4 and R's value to be considered is 2 and we put R = 2 in the formula, we get $(2 \times 2 + 0) \bmod 4$

i.e. $(4 + 0) \bmod 4$

i.e. $4 \bmod 4$

i.e. $4\,\%\,4$

And for the operation 4%4, we get remainder 0 as output and since remainder 0 is assigned to state q_0. Therefore in the above table, state q_2 over input '0' gives state q_0 as output.

*→ Similarly now suppose we are at state q_2 and we receive '1' as input. Then the formula applied here will be (2R + 1) mod 4 and R's value to be considered is 2 and we put R = 2 in the formula, we get $(2 \times 2 + 1) \bmod 4$.

i.e. $(4 + 1) \bmod 4$

i.e. $5 \bmod 4$

i.e. $5\,\%\,4$

And for the operation 5%4, we get remainder 1 as output and since remainder 1 is assigned to state q_1. Therefore in the above table, state q_2 over input '1' gives state q_1 as output.

- Since remainder 3 is assigned to state q_3. Therefore while applying the formula to state q_3, R's value in the formula will be considered as 3.

*→ Now suppose we are at state q_3 and we receive '0' as input. Then the formula applied here will be (2R + 0) mod 4 and R's value to be considered is 3 and we put R = 3 in the formula, we get $(2 \times 3 + 0) \bmod 4$

i.e. $(6 + 0) \bmod 4$

i.e. $6 \bmod 4$

i.e. $6\,\%\,4$

And for the operation 6%4, we get remainder 2 as output and since remainder 2 is assigned to state q_2. Therefore in the above table, state q_3 over input '0' gives state q_2 as output.

*→ Similarly now suppose we are at state q_3 and we receive '1' as input. Then the formula applied here will be (2R + 1) mod 4 and R's value to be considered is 3 and we put R =3 in the formula, we get $(2 \times 3 + 1) \bmod 4$

i.e. $(6 + 1) \bmod 4$

i.e. $7 \bmod 4$

i.e. $7\,\%\,4$

And for the operation 7%4, we get remainder 3 as output and remainder 3 is assigned to state q_3. Therefore in the above table, state q_3 over input '1' gives state q_3 as output.

2. MAF : SXI → 0

S/I	0	1
$\rightarrow q_s$	Y	N
q_0^*	Y	N
q_1	N	N
q_2	Y	N
q_3	N	N

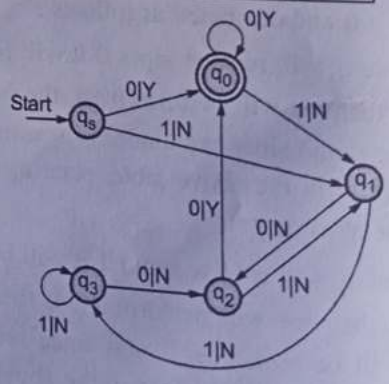


Fig. Ex. 1.1.3(a)

▶ **Step 4 :** Example

$(q_s, 0100)$

$(q_0, 100)$

$(q_1, 00)$

$(q_2, 0)$

$(q\ 0) \rightarrow Y$

UEx. 1.1.4 (MU-Q. 2(a), Dec. 17,Q.2(a), Dec. 18, 10 Marks)

Design FSM to check whether the given ternary number is divisible by 5.

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

▶ **Step 2 :** Logic

$$S = \{q_s, q_0, q_1, q_2, q_3, q_4\}$$

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

➡ **Note :** Here, the input is a ternary number, therefore input is either 0, 1 or 2.

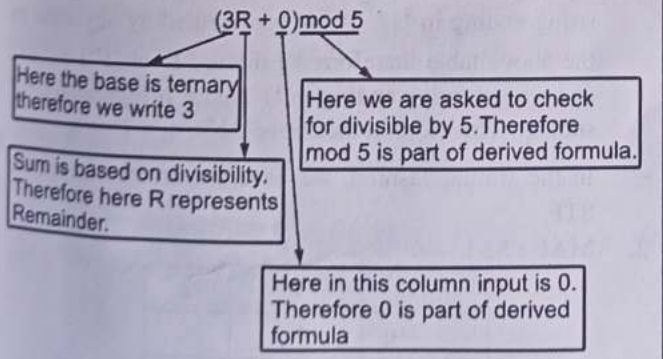
$$O = \{Y, N\}$$

▶ **Step 3 :** Implementation

(1) STF : SXI → S

In this sum, the number system is ternary therefore here we will have to form a dynamic formula in following way.

For the column where input is 0, the formula applied is **(3R + 0) mod 5** and it is derived as follows :



(1A52)**Fig. Ex. 1.1.4**

Similarly, for the column where input is 1 and for the column where input is 2, the formula's applied are **(3R + 1) mod 5** and **(3R + 2) mod 5** respectively.

		(3R + 0) mod 5	(3R + 1) mod 5	(3R + 2) mod 5
	S/I	0	1	2
R	$\rightarrow q_s$	q_0	q_1	q_2
0	q_0^*	q_0	q_1	q_2
1	q_1	q_3	q_4	q_0
2	q_2	q_1	q_2	q_3
3	q_3	q_4	q_0	q_1
4	q_4	q_2	q_3	q_4

2. MAF : SXI → 0

S/I	0	1	2
$\rightarrow q_s$	Y	N	N
q_0^*	Y	N	N
q_1	N	N	Y
q_2	N	N	N
q_3	N	Y	N
q_4	N	N	N

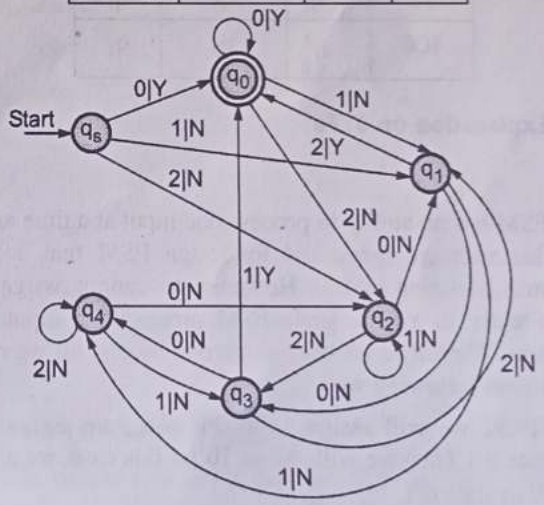


Fig. UEx. 1.1.4(a)

▶ **Step 4 :** Example

$(q_s, 2102)$

$(q_2, 102)$

$(q_2, 02)$

$(q_1, 2)$

$(q\ 0) \rightarrow y$

Ex. 1.1.5 : Design FSM in which input is valid if it ends in "100" over $\sum = \{0, 1\}$.

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

▶ **Step 2 :** Logic

$S = \{q_s, q_0, q_1, q_2, q_3\}$

$I = \{0, 1\}$

$O = \{Y, N\}$

▶ **Step 3 :** Implementation

1. **STF :** SXI → S

S/I		0	1
endsin	$\rightarrow q_s$	q_0	q_1
0	q_0	q_0	q_1
1	q_1	q_2	q_1
10	q_2	q_3	q_1
100	q_3^*	q_0	q_1

☞ **Explanation on STFs**

Note

- FSM has an ability to process one input at a time and in this sum we are asked to design FSM that accepts strings ending in 100. However we cannot assign 100 directly to a state since FSM process one input at a time. Therefore we will assign one by one till we reach 100 in following way.
- FIRST we will assign 1 (in this case, we assign 1 to state q_1) Then we will assign 10 (in this case, we assign 10 to state q_2).
- Then finally we will assign 100 (in this case, we assign 100 to state q_3).

Note : In sums of ends in we need to assign inputs also to states. Therefore here our inputs are 0 and 1 and 1 is assigned to state q_1 and 0 is assigned to state q_0.

Note

- STF table is created with following logic.
- Suppose we are at q_s. q_s is a start state and there is no input already present at start state and we receive input 0 and 0 end in 0, strings ending in 0 is represented by state q_0. Therefore in the above table, $q_s \times 0 \rightarrow q_0$
- Similarly we are at state q_s and we receive input 1, we know 1 ends in 1 and strings ending in 1 are represented by state q_1.

  Therefore in the above table, $q_s \times 1 \rightarrow q_1$
- Now suppose we are at state q_0 and we know that state q_0 represents string ending in 0 and we receive input 0. Therefore now string will be ending in 00 and since no state represents string ending in 00 in the above table. Therefore we start discarding the letter from front till we reach a string which matches with any of string represented by the states In this case, we will discard the 0 from the front '~~0~~0' and we get string ending in '0' which is represented by state q_0.

  Therefore in the above table, $q_0 \times 0 \rightarrow q_0$
- Similarly now we are at state q_0 and we receive input 1, therefore the string becomes ending in '01' (since state q_0 already has a string ending with 0 present). However string ending in 01 is not represented by any state in the above table. Therefore here we discard the front 0 i.e. '~~0~~1' and we get string ending in '1' which is represented by state q_1. Therefore in the above table, $q_0 \times 1 \rightarrow q_1$
- Now suppose we are at state q_1 and state q_1 has already a string ending in 1 present and we receive '0' as input therefore string becomes ending in '10' and string ending in '10' is represented by state q_2.

  Therefore in the above table, $q_1 \times 0 \rightarrow q_2$
- Similarly now we are state q_1 and we receive '1' as input, therefore string becomes ending in '11' however string ending in '11' is not represented by any state in the above table therefore we discard front '1' i.e. '~~1~~1' and we get string ending in '1' and it is represented by state q_1. Therefore in the above table, $q_1 \times 1 \rightarrow q_1$
- In the similar fashion, we have constructed the entire STF.

2. **MAF :** SXI → 0

S/I	0	1
$\rightarrow q_s$	N	N
q_0	N	N
q_1	N	N
q_2	Y	N
q_3^*	N	N

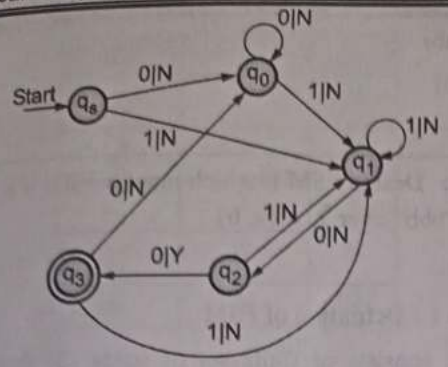


Fig. Ex. 1.1.5

▶ **Step 4 :** Examples

$(q_s, 10100)$

$(q_1, 0100)$

$(q_2, 100)$

$(q_1, 00)$

$(q_2, 0)$

$(q_3) \rightarrow y$

Ex. 1.1.6 : Design FSM in which input is valid if it ends in "babb" over $\sum = \{a, b\}$

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

▶ **Step 2 :** Logic

$S = \{q_s, q_0, q_1, q_2, q_3, q_4\}$

$I = \{a, b\}$

$O = \{Y, N\}$

▶ **Step 3 :** Implementation

1. **STF :** SXI → S

S/I		a	b
endsin	$\rightarrow q_s$	q_0	q_1
a	q_0	q_0	q_1
b	q_1	q_2	q_1
ba	q_2	q_0	q_3
bab	q_3	q_2	q_4
babb	q_4^*	q_2	q_1

2. **MAF :** SXI → O

S/I	a	b
$\rightarrow q_s$	N	N
q_0	N	N
q_1	N	N
q_2	N	N
q_3	N	Y
q_4^*	N	N

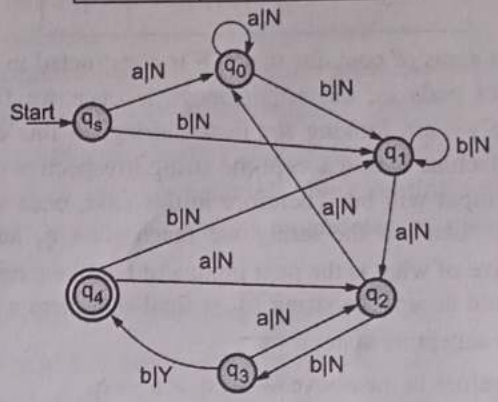


Fig. Ex. 1.1.6

▶ **Step 4 :** Examples

$(q_s, ababb)$

$(q_0, babb)$

(q_1, abb)

(q_2, bb)

(q_3, b)

$(q_4) \rightarrow y$

Ex. 1.1.7 : Design FSM in which input is valid if it Contains "bba" over $\sum = \{a, b\}$

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. **State Function :** STF : S X I → S
2. **Machine Function :** MAF : S X I → O

▶ **Step 2 :** Logic

$S = \{q_s, q_0, q_1, q_2, q_3\}$

$I = \{a, b\}$

$O = \{Y, N\}$

▶ **Step 3 :** Implementation

1. **STF :** SXI → S

S/I		a	b
contains	→ q_s	q_0	q_1
a	q_0	q_0	q_1
b	q_1	q_0	q_2
bb	q_2	q_3	q_2
bba	q_3^*	q_3	q_3

Note : In sums of contains the STF is constructed in similar fashion of ends in, the only change is once we find the substring we are looking for in the string, in this case its "bba". Machine would accept the string irrespective of what the next input will be. Therefore in this case, once we find substring "bba" in the string we reach state q_3 and then irrespective of what is the next input a or b still we remain in state q_3 and accept the string (q_3 is final state here and final states are accepting states).

Therefore in the above table, $q_3 \times a \rightarrow q_3$

and $q_3 \times b \rightarrow q_3$

2. **MAF :** SXI → O

S/I	a	b
→q_s	N	N
q_0	N	N
q_1	N	N
q_2	Y	N
q_3^*	Y	Y

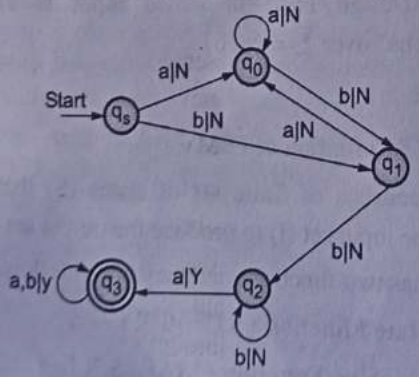


Fig. Ex. 1.1.7

▶ **Step 4 :** Example

(q_s, abbab)

(q_0, bbab)

(q_1, bab)

(q_2, ab)

(q_3, b)

(q_3) → y

Ex. 1.1.8 : Design FSM is which input is valid if it does not contains "bbb" over $\sum = \{a, b\}$

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

▶ **Step 2 :** Logic

$S = \{q_s, q_0, q_1, q_2, q_3\}$

$I = \{a, b\}$

$O = (Y, N)$

▶ **Step 3 :** Implementation

1. **STF :** SXI → S

S/I		a	b
Does not contain	→q_s^*	q_0	q_1
a	q_0^*	q_0	q_1
b	q_1^*	q_0	q_2
bb	q_2^*	q_0	q_3
bbb	q_3	q_3	q_3

Note : In sums of does not contains the STF is constructed in similar fashion of ends in, the only change is once we find the substring we are not looking for in the string, in this case its "bbb". Machine would reject the string. Irrespective of what the next input will be therefore in this case, once we find substring "bbb" in the string we reach state q_3 and then irrespective of what is the next input a or b, still we remain in the state q_3 and reject the string (q_3 is non-final state here and non-final states are rejecting states). Therefore in the above table, $q_3 \times a \rightarrow q_3$,

$q_3 \times b \rightarrow q_3$

Also note here we will reject the strings if and only if we found substring "bbb" in it otherwise any other string will be accepted. Therefore in the above table, leaving state q_3 all other states are final states i.e. accepting states.

2. **MAF :** SXI → O

S/I	a	b
$\rightarrow q_s^*$	Y	Y
q_0^*	Y	Y
q_1^*	Y	Y
q_2^*	Y	N
q_3	N	N

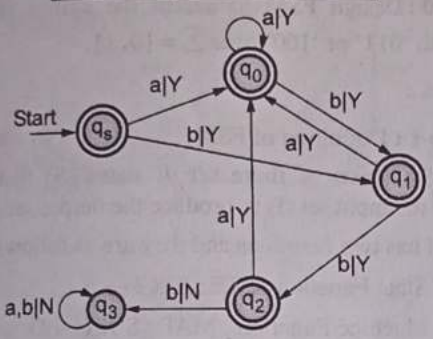


Fig. Ex. 1.1.8

▶ **Step 4 :** Example

(q_s, abbbb)

(q_0, bbbb)

(q_1, bbb)

(q_2, bb)

(q_3, b)

(q_3) → N

Ex. 1.1.9 : Design FSM to accept the strings if it starts with three consecutive as over $\sum = \{a, b\}$.

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

▶ **Step 2 :** Logic

$S = \{q_s, q_0, q_1, q_2, q_3\}$

$I = \{a, b\}$

$O = \{Y, N\}$

▶ **Step 3 :** Implementation

1. **STF :** SXI → S

S/I		a	b
A → q_s		q_0	q_3
a	q_0	q_1	q_3
aa	q_1	q_2	q_3
aaa	q_2^*	q_2	q_2
	q_3	q_3	q_3

Note : In this sum, we need to design a FSM which accept strings that start with three consecutive a's and after that any combination between a and b can follow. So here we have constructed STF in following manner :

- Now suppose we are at start state q_s and we receive 'a' as input, then we will have string starting with 'a' and string starting with 'a' is represented by state q_0 in the above table.

  Therefore in the above table, $q_s \times a \rightarrow q_0$
- Similarly now we are at start state q_s and

  We receive 'b' as input, then we will have string starting with 'b'. Therefore we are going to reject this string even without scanning or processing the entire string as in this sum our machine will only accept strings which start with three consecutive a's. Therefore we are going to put this string in dead state and dead state is represented by state q_3 in the above table.

  Therefore in the above table, $q_s \times b \rightarrow q_3$
- Now suppose we are at state q_0 and since we are at q_0 'a' is already present at q_0 and now we receive 'a' as input and therefore we have a string starting with two consecutive a's i.e. 'aa' and string starting with two consecutive a's is represented by state q_1 in the above table. Therefore in the above table, $q_0 \times a \rightarrow q_1$
- Now similarly suppose we are at state q_0 and we receive 'b' as input. Therefore we have a string starting with 'ab' and in this sum we need to accept strings starting with three consecutive a's. Therefore we will reject this string without even further processing the entire string and put it into a dead state and dead state is represented by state q_3 in the above table.

  Therefore in the above table, $q_0 \times b \rightarrow q_3$
- Now suppose we are at state q_1 and since we are at q_1 'aa' is already present at q_1 and now we receive 'a' as input and therefore we have a string starting with three

consecutive a's i.e. 'aaa' and string starting with three consecutive a's is represented by state q_2 in the above table.

Therefore in the above table, $q_1 \times a \rightarrow q_2$

- Similarly now suppose we are state q_1 and we receive 'b' as input, then the string will become starting with 'aab' and in this sum we accept only those strings that start with three consecutive a's , therefore we will reject this string and put it into a dead state and in the above table dead state is represented by state q_3.

  Therefore in the above table, $q_1 \times b \rightarrow q_3$

- Now suppose we are at state q_2 and therefore we already have 'aaa' present, since three consecutive a's are already present at q_2. Therefore it implies string is already starting with three consecutive a's i.e. aaa. Therefore irrespective of what is the next input a or b, we are going to accept this string and q_2 is final state (Final states are accepting state).

  Therefore in the above table, $q_2 \times a \rightarrow q_2$ and $q_2 \times b \rightarrow q_2$

- Now suppose we are at state q_3 and in the above table q_3 is a dead state i.e. it is a rejecting state. Therefore once we reach state q_3, irrespective of what will be the next input a or b we are going to still reject the string, that means we will be in state q_3 only. Therefore in the above table, $q_3 \times a \rightarrow q_3$ and $q_3 \times b \rightarrow q_3$.

2. MAF : SXI → O

S/I	a	b
$\rightarrow q_s$	N	N
q_0	N	N
q_1	Y	N
q_2^*	Y	Y
q_3	N	N

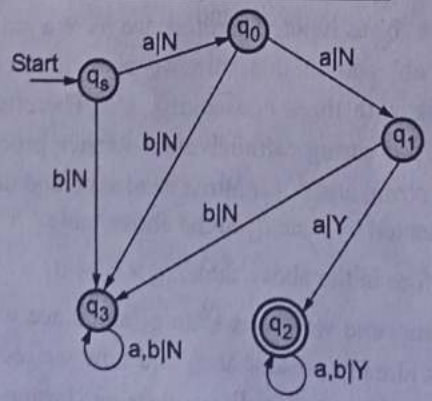


Fig. Ex. 1.1.9

Step 4 : Example

$(q_s, aaaba)$

$(q_0, aaba)$

(q_1, aba)

(q_2, ba)

(q_2, a)

$(q_2) \rightarrow y$

Ex. 1.1.10 : Design FSM to accept the strings if it starts either with '011' or '100' over $\sum = \{0, 1\}$.

☑ **Soln. :**

Step 1 : Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

Step 2 : Logic

$S = \{q_s, q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$

$I = \{0, 1\}$

$O = \{Y, N\}$

Step 3 : Implementation

1. STF : SXI → S

S/I		0	1
$A \rightarrow q_s$		q_0	q_3
0	q_0	q_6	q_1
01	q_1	q_6	q_2
011	q_2^*	q_2	q_2
1	q_3	q_4	q_6
10	q_4	q_5	q_6
100	q_5^*	q_5	q_5
	q_6	q_6	q_6

2. MAF : SXI → O

S/I	0	1
$\rightarrow q_s$	N	N
q_0	N	N
q_1	N	Y

S/I	0	1
q_2^*	Y	Y
q_3	N	N
q_4	Y	N
q_5^*	Y	Y
q_6	N	N

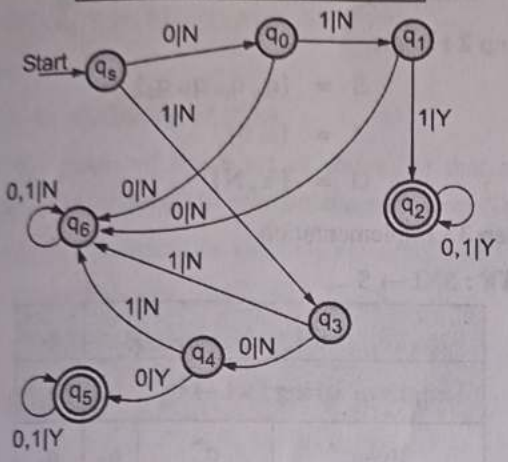


Fig. Ex. 1.1.10

▶ **Step 4 :** Example

$(q_s, 1001)$

$(q_3, 001)$

$(q_4, 01)$

$(q_5, 1)$

$(q_5) \rightarrow Y$

Ex. 1.1.11 : Design FSM to accept the strings if it contains 'a' at every even position over $\sum = \{a, b\}$.

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

▶ **Step 2 :** Logic

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

$I = \{a, b\}$

$O = \{Y, N\}$

▶ **Step 3 :** Implementation

1. STF : SXI → S

S/I		a	b
Length of string | x | → q_s^*		q_1	q_1
Even	q_0^*	q_1	q_1
Odd	q_1^*	q_0	q_2
	q_2	q_2	q_2

Note

- In this sum, we are asked to design FSM that accepts strings in which at every even position is 'a' over $\sum = \{a, b\}$. Here our logic is based on length of string and length of string can be either even or odd.
- Here, even length of string is represented by state q_0 and odd length of string is represented by state q_1 and q_2 is a dead state i.e. rejecting state and STF is constructed in following manner.
- Now suppose we are at start state q_s, since q_s is a start state, therefore there is no input already present at q_s and since there is no input already present therefore whichever input comes at start state it will come on first position and first position is an odd position and in this sum, there is no condition applied on odd position. (i.e. both a and b can come at odd position).
- That means here both a and b can come as input and therefore either a comes as input or b comes as input, length of string will become 1 i.e. odd and odd length of string is represented by state q_1 in the above table. Therefore in the above table, $q_s \times a \rightarrow q_1$ and $q_s \times b \rightarrow q_1$
- Now suppose we are at state q_0 and since we are at state q_0 there is a string of even length already present. Therefore the next input will come on odd position and in this sum there is no condition on odd position. Therefore both a and b can a come as input and either 'a' comes as input or 'b' comes as input, length of string will be odd and odd length of string is represented by state q_1 in the above table. Therefore in the above table, $q_0 \times a \rightarrow q_1$ and $q_0 \times b \rightarrow q_1$
- Now suppose we are at state q_1 and since we are at state q_1 there is a string of odd length already present. Therefore the next input will come. on even position and in this sum there is condition on even position i.e. only a's can come on even position.

  *→ Now suppose we are at state q_1 and 'a' comes as input and since 'a' can come as input at even position and if a comes as input then length of string will

become even and even length of string is represented by state q_0 in the above table.

Therefore in the above table, $q_1 \times a \rightarrow q_0$

- *→ Now suppose, we are at state q_1 and we receive 'b' as input and since b cannot come at even position. Therefore we are going to reject this string that means put it to dead state and here q_2 represents the dead state in the above table.

Therefore in the above table, $q_1 \times b \rightarrow q_2$

- Now suppose we are at state q_2. Since state q_2 is rejecting state. Therefore either 'a' comes or 'b' comes as a next input still we will be in state q_2.
- Therefore in the above table, $q_2 \times a \rightarrow q_2$ & $q_2 \times b \rightarrow q_2$

2. MAF : SXI → O

S/I	a	b
$\rightarrow q_s^*$	Y	Y
q_0^*	Y	Y
q_1^*	Y	N
q_2	N	N

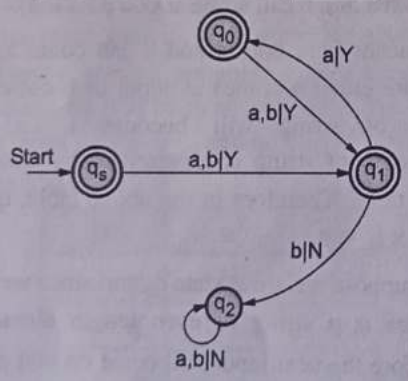


Fig. Ex. 1.1.11

▶ **Step 4 :** Example

$(q_s, baab)$

(q_1, aab)

(q_0, ab)

(q_1, b)

$(q_2) \rightarrow N$

Ex. 1.1.12 : Design FSM to accept strings if it contains 'a' at every even position and 'b' at every odd position over $\sum = \{a, b\}$.

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

▶ **Step 2 :** Logic

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

$I = \{a, b\}$

$O = \{Y, N\}$

▶ **Step 3 :** Implementation

1. STF : SXI → S

S/I		a	b
Length of string | x | → q_s^*		q_2	q_1
Even	q_0^*	q_2	q_1
Odd	q_1^*	q_0	q_2
	q_2	q_2	q_2

2. MAF : SXI → O

S/I	a	b
$\rightarrow q_s^*$	N	Y
q_0^*	N	Y
q_1^*	Y	N
q_2	N	N

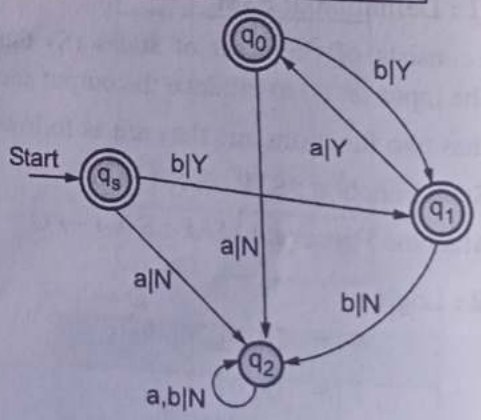


Fig. Ex. 1.1.12

▶ **Step 4 :** Example

$(q_s, baba)$

(q_1, aba)

(q_0, ba)

(q_1, a)

$(q_0) \rightarrow Y$

Ex. 1.1.13 : Design FSM to accept the strings if it contains exactly 3a's $\Sigma = \{a, b\}$.

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

▶ **Step 2 :** Logic

$S = \{q_s, q_0, q_1, q_2, q_3, q_4\}$

$I = \{a, b\}$

$O = \{Y, N\}$

▶ **Step 3 :** Implementation

S/I		a	b
No of a's	$\rightarrow q_s$	q_1	q_0
0	q_0	q_1	q_0
1	q_1	q_2	q_1
2	q_2	q_3	q_2
3	q_3^*	q_4	q_3
More than 3	q_4	q_4	q_4

Note

- In this sum, we need to design a FSM which will accept the strings that contains exactly three a's. So here strings should have exactly three a's, not necessarily they should be consecutive and there can be any number of b's as there is no condition on b's. As we know FSM has ability to process one input at a time. So therefore machine directly cannot recognize 3a's in the string rather it will recognize it step by step in following manner :
- At the beginning, there will be 0 a's in the string, 0 a's will be assigned to or represented by state q_0.
- Similarly then 1a's will be represented by state q_1.
- Similarly then 2a's will be represented by state q_2.
- Similarly then 3a's will be represented by state q_3.
- Similarly more than 3a's will be represented by state q_4.

Now the STF is constructed in following way :

- Now suppose we are at start state q_s and we receive 'a' as an input. Then the number of a's in the string would become 1 as we do not have any string already present at the start state q_s and 1a's is represented by state q_1 in the above table.

  Therefore in the above table, $q_s \times a \rightarrow q_1$
- Similarly now we are at start state q_s and we receive 'b' as an input, therefore number of a's in the string will be 0 and 0 a's are represented by state q_0 in the above table.

  Therefore in the above table, $q_s \times b \rightarrow q_0$
- Now suppose we are at state q_0, therefore we already have 0 a's present and we receive 'a' as an input, therefore the number of a's in the string will be 1 and 1 a's are represented by state q_1 in the above table.

  Therefore in the above table, $q_0 \times a \rightarrow q_1$
- Similarly now we are at state q_0 and we receive 'b' as an input, therefore the number of a's in the string will be still 0 and 0 a's are represented by state q_0 in the above table.

  Therefore in the above table, $q_0 \times b \rightarrow q_0$
- Now suppose we are at state q_1 and therefore we have '1a' already present and we receive 'a' as an input therefore the number of a's in the string will be 2 and 2a's are represented by state q_2 in the above table. Therefore in the above table, $q_1 \times a \rightarrow q_2$
- Similarly, now we are at state q_1 and we receive 'b' as an input, therefore the number of a's in the string will be still 1 and 1a's are represented by state q_1 in the above table. Therefore in the above table, $q_1 \times b \rightarrow q_1$
- Now suppose we are at state q_2 and therefore we have 2a's already present and we receive 'a' as an input, therefore the number of a's in the string will become 3 and 3a's are represented by state q_3 in the above table.

  Therefore in the above table, $q_2 \times a \rightarrow q_3$
- Similarly now we are at state q_2 and we receive 'b' as an input, therefore the number of a's in the string still will be 2 and 2a's are represented by state q_2 in the

above table. Therefore in the above table, $q_2 \times b \rightarrow q_2$

- Now suppose we are at state q_3, therefore we already have 3a's present and we receive 'a' as an input, therefore the number of a's in the string will become more than 3 and more than 3a's are represented by state q_4 in the above table.

  Therefore in the above table, $q_3 \times a \rightarrow q_4$
- Similarly now we are at state q_3 we receive 'b' as a input. Therefore the number of a's in the string will still be 3 and 3a's are represented by state q_3 in the above table.Therefore in the above table, $q_3 \times b \rightarrow q_3$
- Now suppose we are at state q_4, therefore we already have more than 3a's and therefore whatever is the next input 'a' or 'b', still we will have more than 3a's in the string and more than 3a's are represented by state q_4 in the above table.

  Therefore in the above table, $q_4 \times a \rightarrow q_4$

  and $q_4 \times b \rightarrow q_4$

Note : Here, we need to accept strings which have exactly 3a's therefore q_3 state will be final state here as it represents strings having exactly 3a's.

MAF : SXI → O

S/I	a	b
$\rightarrow q_s$	N	N
q_0	N	N
q_1	N	N
q_2	Y	N
q_3^*	N	Y
q_4	N	N

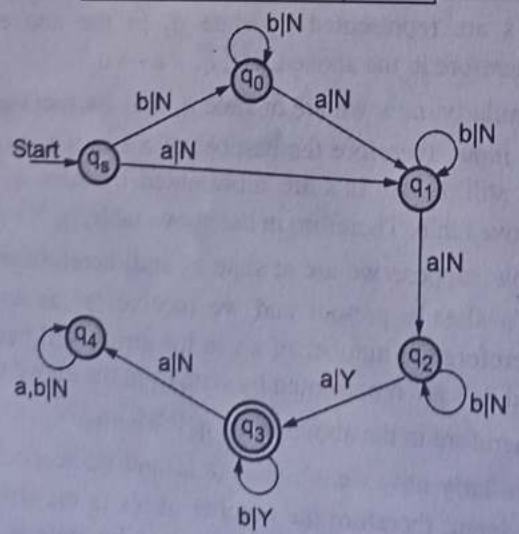


Fig. Ex. 1.1.13

Step 4 : Example

$(q_s, baaa)$

(q_0, aaa)

(q_1, aa)

(q_2, a)

$(q_3) \rightarrow Y$

Ex. 1.1.14 : Design FSM to accept the strings if it contains at the most 3a's over $\sum = \{a, b\}$.

Soln. :

Step 1 : Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

Step 2 : Logic

$S = \{q_s, q_0, q_1, q_2, q_3, q_4\}$

$I = \{a, b\}$

$O = \{Y, N\}$

Step 3 : Implementation

1. **STF :** SXI → S

S/I		a	b
No of a's	$\rightarrow q_s^*$	q_1	q_0
0	q_0^*	q_1	q_0
1	q_1^*	q_2	q_1
2	q_2^*	q_3	q_2
3	q_3^*	q_4	q_3
More than 3		q_4	q_4	q_4

Note

In this sum, we need to design a FSM which accepts strings that have at the most 3a's. Therefore the STF construction in this sum is similar to STF construction as in the previous sum, only change is now we need to accept strings that have at most 3a's (i.e. 0, 1, 2 and 3a's in it) rather than just accepting strings that have exactly 3a's. Therefore here final states will be q_s, q_0, q_1, q_2 and q_3.

2. **MAF :** SXI → O

S/I	a	b
$\rightarrow q_s^*$	Y	Y
q_0^*	Y	Y
q_1^*	Y	Y
q_2^*	Y	Y
q_3^*	N	Y
q_4	N	N

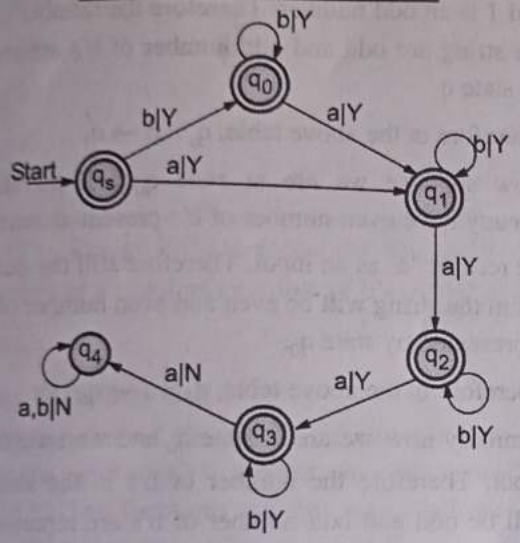


Fig. Ex. 1.1.14

▶ **Step 4 :** Example

(q_s, baba)

(q_0, aba)

(q_1, ba)

(q_1, a)

(q_2) → Y

Ex. 1.1.15 : Design FSM to accept the strings if it contains at least 3a's over $\sum = \{a, b\}$.

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

▶ **Step 2 :** Logic

$$S = \{q_s, q_0, q_1, q_2, q_3, q_4\}$$

$$I = \{a, b\} \; ; \quad O = \{Y, N\}$$

▶ **Step 3 :** Implementation

1. **STF :** SXI → S

S/I		a	b
No of a's	$\rightarrow q_s$	q_1	q_0
0	q_0	q_1	q_0
1	q_1	q_2	q_1
2	q_2	q_3	q_2
3	q_3^*	q_4	q_3
More than 3	q_4^*	q_4	q_4

Note

- In this sum, we need to design a FSM which accepts strings that have atleast 3a's. Therefore here the STF will be similar to STF in sum of exactly 3a's.
- However the only change is here we need to accept strings that have atleast 3a's in it. i.e. strings having 3 or more than 3a's will be accepted. Therefore here the final states will be q_3 and q_4.

2. **MAF :** SXI → O

S/I	a	b
$\rightarrow q_s$	N	N
q_0	N	N
q_1	N	N
q_2	Y	N
q_3^*	Y	Y
q_4^*	Y	Y

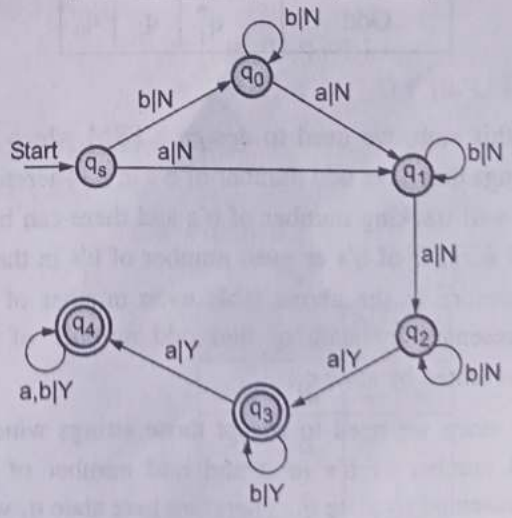


Fig. Ex. 1.1.15

▶ **Step 4 :** Example

$(q_s, babaaa)$

$(q_0, abaaa)$

$(q_1, baaa)$

(q_1, aaa)

(q_2, aa)

(q_3, a)

$(q_4) \rightarrow Y$

Ex. 1.1.16 : Design FSM to accept the strings if it contains odd number of b's over $\Sigma = \{a, b\}$.

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. **State Function :** STF : S X I → S
2. **Machine Function :** MAF : S X I → O

▶ **Step 2 :** Logic

$S = \{q_s, q_0, q_1\}$

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

$O = \{Y, N\}$

▶ **Step 3 :** Implementation

1. **STF :** SXI → S

S/I		a	b
No of b's	$\rightarrow q_s$	q_0	q_1
Even	q_0	q_0	q_1
Odd	q_1^*	q_1	q_0

Note

- In this sum, we need to design a FSM which accepts strings that have odd number of b's in it. Therefore here we will tracking number of b's and there can be either odd number of b's or even number of b's in the string. Therefore in the above table even number of b's are represented by state q_0 and odd number of b's are represented by state q_1.

  and since we need to accept those strings which have odd number of b's in it and odd number of b's are represented by state q_1. Therefore here state q_1 will be a final state.

- For this sum, STF is constructed in following manner :

  Now suppose we are at state q_s and we receive 'a' as an input and since at start state there is no input already present. Therefore the number of b's in the string will be 0 and 0 is an even number. Therefore number of b's in the string are even and even number of b's are represented by state q_0. Therefore in the above table, $q_s \times a \rightarrow q_0$

- Similarly now we are at state q_s and we receive 'b' as an input. Hence the number of b's in the string will be 1 and 1 is an odd number. Therefore the number of b's in the string are odd and odd number of b's are represent by state q_1.

  Therefore in the above table, $q_s \times b \rightarrow q_1$

- Now suppose we are at state q_0 and therefore we already have even number of b's present at state q_0 and we receive 'a' as an input. Therefore still the number of b's in the string will be even and even number of b's are represented by state q_0.

  Therefore in the above table, $q_0 \times a \rightarrow q_0$

- Similarly now we are at state q_0 and we receive 'b' as input. Therefore the number of b's in the string now will be odd and odd number of b's are represented by state q_1.

  Therefore in the above table, $q_0 \times b \rightarrow q_1$

- Now suppose we are at state q_1 and therefore we already have odd number of b's present at state q_1 and we receive 'a' as input, therefore the number of b's in the string still will be odd and odd number of b's are represented by state q_1. Therefore in the above table, $q_1 \times a \rightarrow q_1$

- Similarly now suppose we are at state q_1 and we receive 'b' as an input. Therefore the number of b's in the string now will become even and even number of b's are represented by state q_0.

  Therefore in the above table, $q_1 \times b \rightarrow q_0$

2. **MAF :** SXI → O

S/I	a	b
$\rightarrow q_s$	N	Y
q_0	N	Y
q_1^*	Y	N

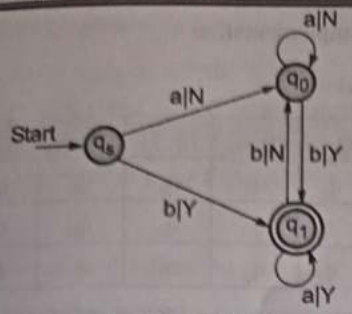


Fig. Ex. 1.1.16

▶ **Step 4 :** Example

(q_s, aba)

(q_0, ba)

(q_1, a)

(q_1) → Y

Ex. 1.1.17 : Design FSM to accept the strings if it contains even number of a's and odd number of b's over $\sum = \{a, b\}$.

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

▶ **Step 2 :** Logic

$S = \{q_s, q_0, q_1, q_2, q_3\}$

$I = \{a, b\}$

$O = \{Y, N\}$

▶ **Step 3 :** Implementation

1. STF : SXI → S

		S/I	a	b
No. of a's	No. of b's	→q_s	q_2	q_1
Even	Even	q_0	q_2	q_1
Even	Odd	q_1^*	q_3	q_0
Odd	Even	q_2	q_0	q_3
Odd	Odd	q_3	q_1	q_2

2. MAF : SXI → O

S/I	a	b
→q_s	N	Y
q_0	N	Y
q_1^*	N	N
q_2	N	N
q_3	Y	N

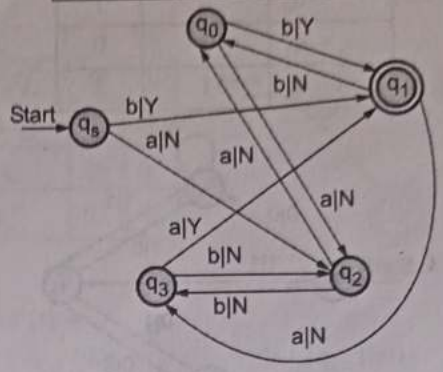


Fig. Ex. 1.1.17

▶ **Step 4 :** Example

(q_s, baa)

(q_1, aa)

(q_3, a)

(q_1) → y

Ex. 1.1.18 : Design FSM to output the remainder when binary number is divided by 3.

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. State Function : STF : S X I → S
2. Machine Function : MAF : S X I → O

▶ **Step 2 :** Logic

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

$I = \{0, 1\}$; $O = \{0, 1, 2\}$

▶ **Step 3 :** Implementation

1. STF : SXI → S

(2R + 0) mod 3

(2R + 1) mod 3

	S/I	0	1
R→	q_s	q_0	q_1
0	q_0	q_0	q_1
1	q_1	q_2	q_0
2	q_2	q_1	q_2

2. **MAF :** SXI → O

S/I	0	1
→q_s	0	1
q_0	0	1
q_1	2	0
q_2	1	2

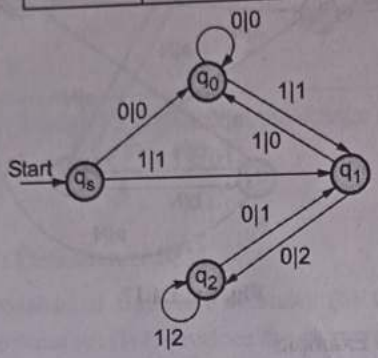


Fig. Ex. 1.1.18

▶ **Step 4 :** Example

(q_s, 011010)
(q_0, 11010)
(q_1, 1010)
(q_0, 010)
(q_0, 10)
(q_1, 0)
(q_2) → 2

Ex. 1.1.19 : Design FSM to implement binary Adder. or Design FSM to add two binary number of same length.

☑ **Soln. :**

▶ **Step 1 :** Definition of FSM

FSM consists of finite set of states (S) that alter on receiving the input set (I) to produce the output set (O).

FSM has two functions and they are as follows :

1. **State Function :** STF : S X I → S
2. **Machine Function :** MAF : S X I → O

▶ **Step 2 :** Logic

S = {q_s, q_0, q_1}
I = {(0, 0), (0, 1), (1, 0), (1, 1)}
O = {0, 1}

▶ **Step 3 :** Implementation

1. **STF :** SXI → S

S/I	(0, 0)	(0, 1)	(1, 0)	(1, 1)
→ q_s	q_0	q_0	q_0	q_1
NC q_0	q_0	q_0	q_0	q_1
C q_1	q_0	q_1	q_1	q_1

'NC' stands for no carry; 'C' stands for carry.

2. **MAF :** SXI → S

S/I	(0, 0)	(0, 1)	(1, 0)	(1, 1)
→ q_s	0	1	1	0
NC q_0	0	1	1	0
C q_1	1	0	0	1

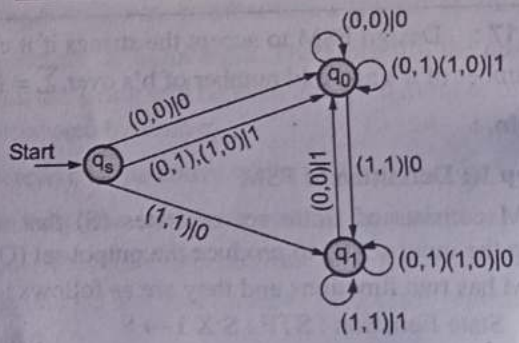


Fig. Ex. 1.1.19

▶ **Step 4 :** Example

(q_s, (0010, 0110)) 1000
(q_0, 001, 011)
(q_1, (00, 01))
(q_1, (0, 01))
(q_0,)

1.1.4 FSM Properties

1. **Periodicity**

A FSM do not have the capacity to remember large amount of information because it has limited number of states and this sets the limit to the length of the sequence it can remember therefore it repeats some state or states again and again, that means some sequence of states will be repeated periodically.