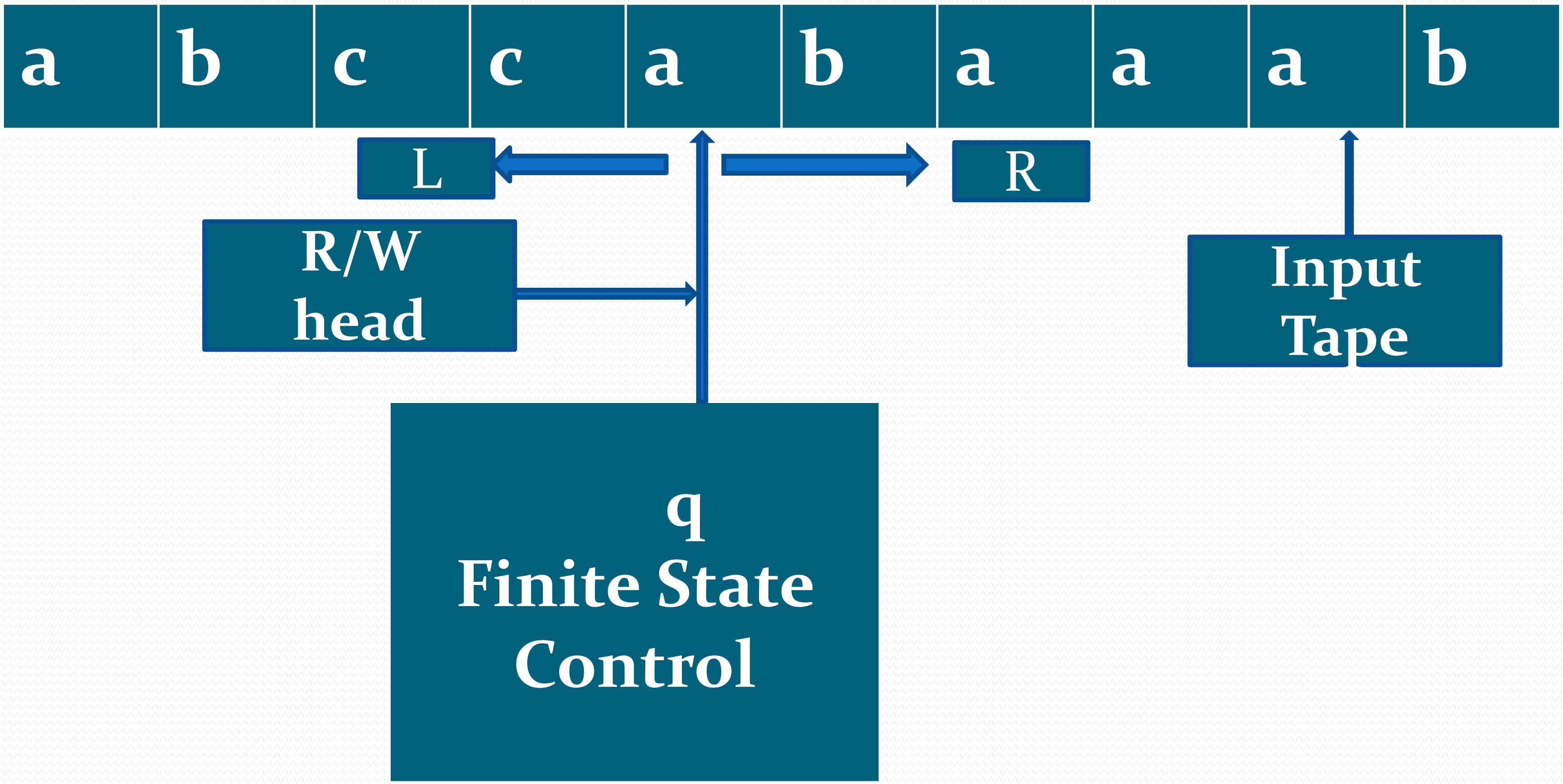
Turing Machine

What is Turing machine?

- A **Turing machine** is a mathematical model of computation that defines an abstract **machine**, which manipulates symbols on a strip of tape according to a table of rules.
- It is a generalized machine which can accept all the type of languages i.e. regular , context free, context sensitive, recursive and recursive enumerable languages .
- There are two purposes for a **Turing machine**: deciding formal languages and solving mathematical functions.

Model of Turing Machine



Mathematical Definition of Turing Machine (TM)

A Turing machine is described by a 7-tuple
$M=(Q, \Sigma, \Gamma, \delta, q_o, B, F)$ where,

- Q is the finite set of states,
- $\Sigma \subseteq \Gamma$ is the set of input symbols
- Γ is the set of tape symbols,
- $q_o \in Q$ is the initial state,
- $B \in \Gamma$ is a blank symbol
- F is the set of final states, and
- δ is a transition function which is defined as following:-
- $\delta: Q\times\Gamma \rightarrow Q\times\Gamma\times\{L, R\}$

where,
L represents left direction and R represents right direction.

Instantaneous Description(ID)

An instantaneous description of TM is a string of the following form:-

$$\alpha \, q \, \beta$$

Where, $\alpha, \beta \in \Gamma^*$, $q \in Q$.

$\alpha\beta$ denotes the whole contents of the tape.

q is a current state.

R/W head of machine will be at the leftmost symbol of β.

Initial ID will be $q_o w$. where $w \in \Sigma^*$

Move relation

This relation exist between two consecutives ID's. It is dented by $\vdash_M$.

Consider an ID of a TM at any instant is

$a_1a_2a_3\ldots\ldots\ldots a_{i-1}qa_ia_{i+1}\ldots\ldots\ldots a_n$.

(1) If $\delta(q, a_i) = (p, y, R)$ then move of machine will be

$a_1a_2a_3\ldots\ldots\ldots a_{i-1}qa_ia_{i+1}\ldots\ldots\ldots a_n \vdash_M a_1a_2a_3\ldots\ldots\ldots a_{i-1}\, ypa_{i+1}\ldots\ldots\ldots a_n$

(2) If $\delta(q, a_i) = (p, y, L)$ then move of machine will be

$a_1a_2a_3\ldots\ldots\ldots a_{i-1}qa_ia_{i+1}\ldots\ldots\ldots a_n \vdash_M a_1a_2a_3\ldots\ldots\ldots pa_{i-1}\, ya_{i+1}\ldots\ldots\ldots a_n$

Language accepted by TM

The language accepted by Turing machine M is defined as following:-

$L(M) = \{ w \,!\, q_o w \vdash_M^* \alpha\, p\, \beta$ where $w \in \Sigma^*$,

$p \in F$ and $\alpha, \beta \in \Gamma^*$ and machine halts on the final state.$\}$

Representation of TM

Two representations are used for TM.

(1) By transition table (2) By transition diagram

By transition table

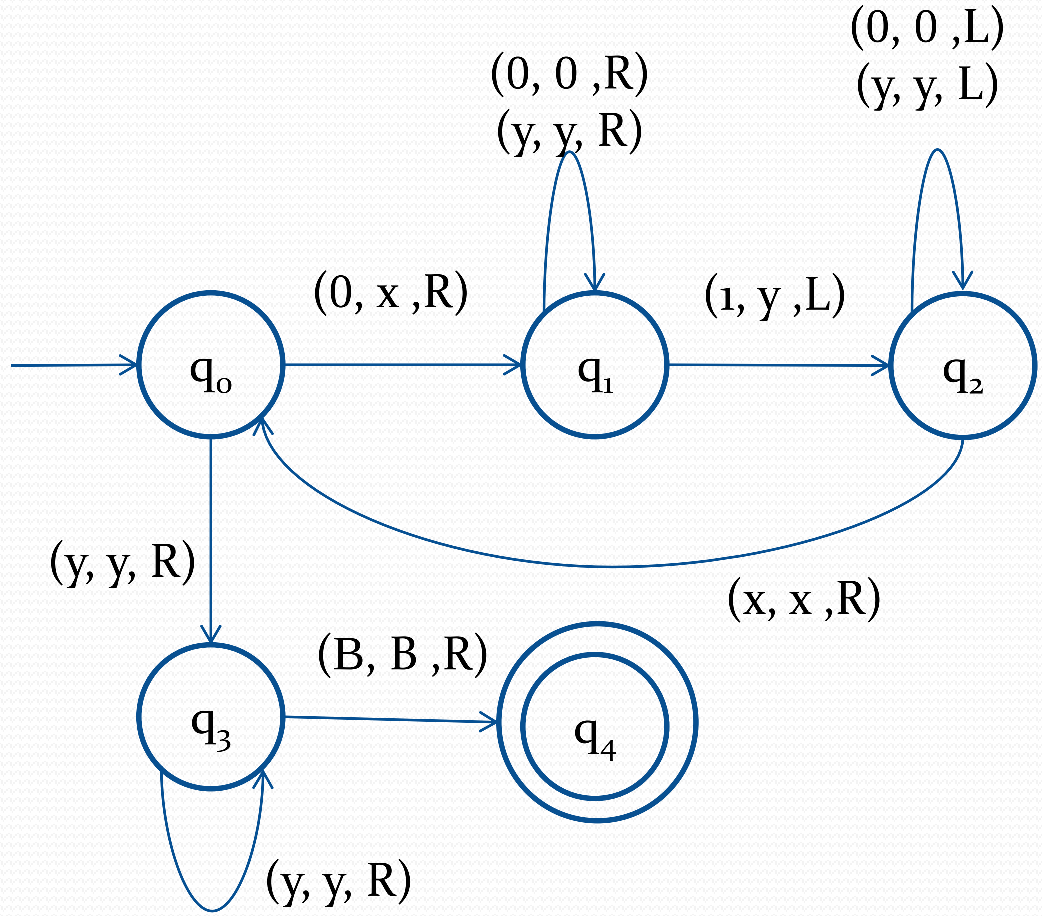
δ	Tape symbols				
States	0	1	x	y	B
$\rightarrow q_0$	(q_1, x, R)			(q_3, y, R)	
q_1	$(q_1, 0, R)$	(q_2, y, L)		(q_1, y, R)	
q_2	$(q_2, 0, L)$		(q_0, x, R)	(q_2, y, L)	
q_3				(q_3, y, R)	(q_4, B, R)
q_4					

Turing Machine

Part-2

Representation of TM(continue)

By transition diagram



Processing or working of TM

Ex. Check the acceptability of following strings

(1) 0011 (2) 011 (3) 00101

by the TM in the previous example.

Solution:

(1) For string 0011

$q_0 0011 \vdash xq_1 011 \vdash x0q_1 11 \vdash xq_2 0y1 \vdash q_2 x0y1 \vdash xq_0 0y1 \vdash xxq_1 y1 \vdash xxyq_1 1 \vdash xxq_2 yy \vdash xq_2 xyy \vdash xxq_0 yy \vdash xxyq_3 y \vdash xxyyq_3 B \vdash xxyyBq_4 B$ (machine halts at final state)

Since machine halts at final state, therefore this string is accepted by TM.

Processing or working of TM(continue)

(2) For string 011

$q_0 011 \vdash xq_1 11 \vdash q_2 xy1 \vdash xq_0 y1 \vdash xyq_3 1$ (machine halts at non-final state)

Since machine halts at non-final state, therefore this string is not accepted by TM.

(3) For string 00101

$q_0 00101 \vdash xq_1 0101 \vdash x0q_1 101 \vdash xq_2 0y01 \vdash q_2 x0y01 \vdash xq_0 0y01 \vdash xxq_1 y01 \vdash xxyq_1 01 \vdash xxy0q_1 1 \vdash xxyq_2 0y \vdash xxq_2 y0y \vdash xq_2 xy0y \vdash xxq_0 y0y \vdash xxyq_3 0y$ (machine halts at non-final state)

Since machine halts at non-final state, therefore this string is not accepted by TM.

Construction of TM

In this section, we shall see how TM's can be constructed.

Ex. Construct TM for the following languages:-

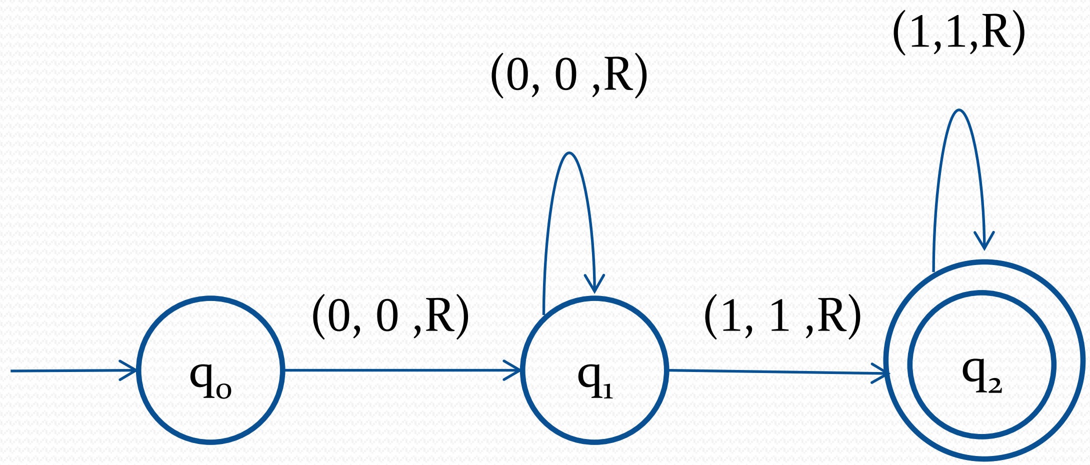
1) $L = \{ 0^m 1^n ! \ m, n \geq 1\}$
2) L= the set of all strings of 0 and 1 which contain 001 as a substring.
3) L= the set of all strings of 0 and 1 ending with 101.
4) L= the set of strings of a and b which contains at least one a's and exactly two b's.

Ex. $L = \{ 0^m 1^n ! \; m, n \geq 1\}$

Solution:

Procedure: First check given language is regular or not. If language is regular then first construct DFA for that language. After, convert it into TM.

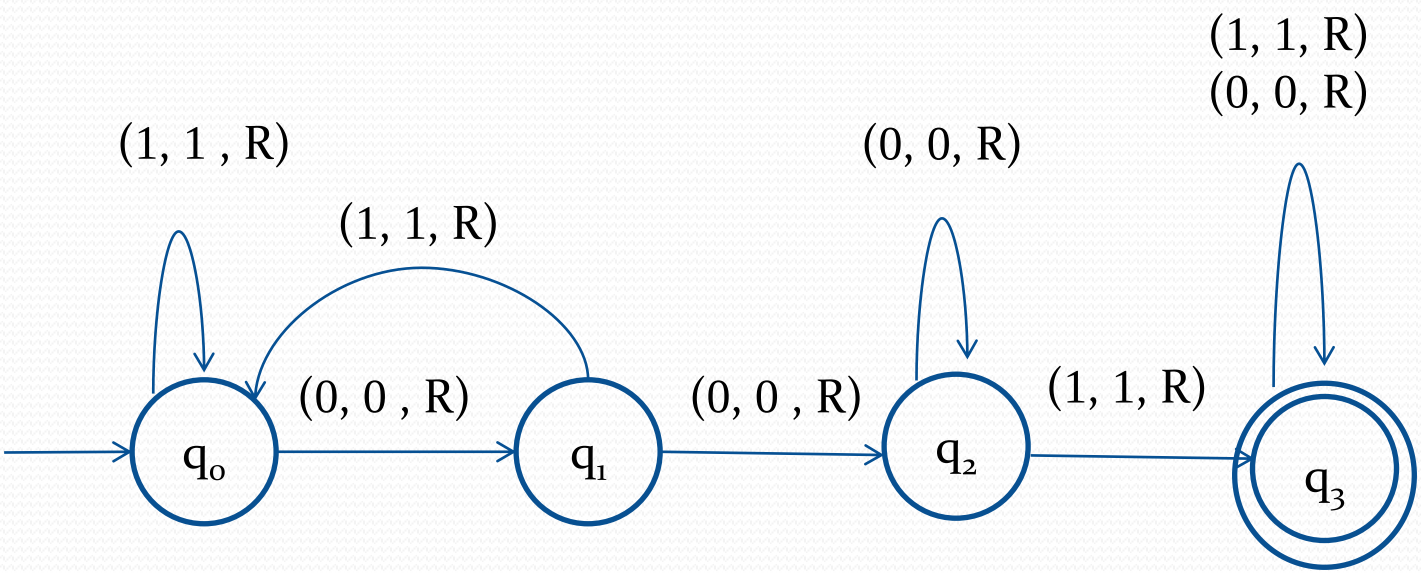
Since this language is regular, Therefore the TM for this language will be



Ex. L= the set of all strings of 0 and 1 which contain 001 as a substring.

Solution:

Since this language is regular, therefore the TM for this language is



Turing Machine

Part-3

Ex. Construct Turing machine for the language

$$L = \{ 0^n 1^n \; ! \; n \geq 1 \}.$$

Solution:

To construct Turing for a language, first we have to identify the pattern of strings belongs in to L. Some strings are 01, 0011, 000111 etc.

Now, you have to think, how machine move from initial ID to final ID.

Procedure: Initially machine starts at the initial state q_o. machine scan the tape string. If the current tape symbol is 0, then machine change its state, replace the current input symbol 0 by another tape symbol and also the head of machine move in the right direction.

Machine move in the right direction until 1 appears in the tape. As soon as 1 appears in the tape, machine replaces 1 by some another tape symbol, return to back i.e. move in the left direction and change its state.

$L = \{ 0^n 1^n \mid n \geq 1 \}$

Machine move in the left direction till leftmost 0 appears in tape. As soon as, machine be reached at leftmost 0, its state becomes q_o.

we repeat the whole process till any 0 in the tape. As soon as, all 0's are deleted from tape, we check number of 1's in tape. If there is any 1's in the tape, then machine reject the string otherwise machine may accept the string.

Therefore, the TM corresponding to this language will be

$M=(\{q_o,q_1,q_2,q_3, q_4\}, \{0, 1\}, \{0, 1, x, y, B\}, q_o, B, \{q_4\})$

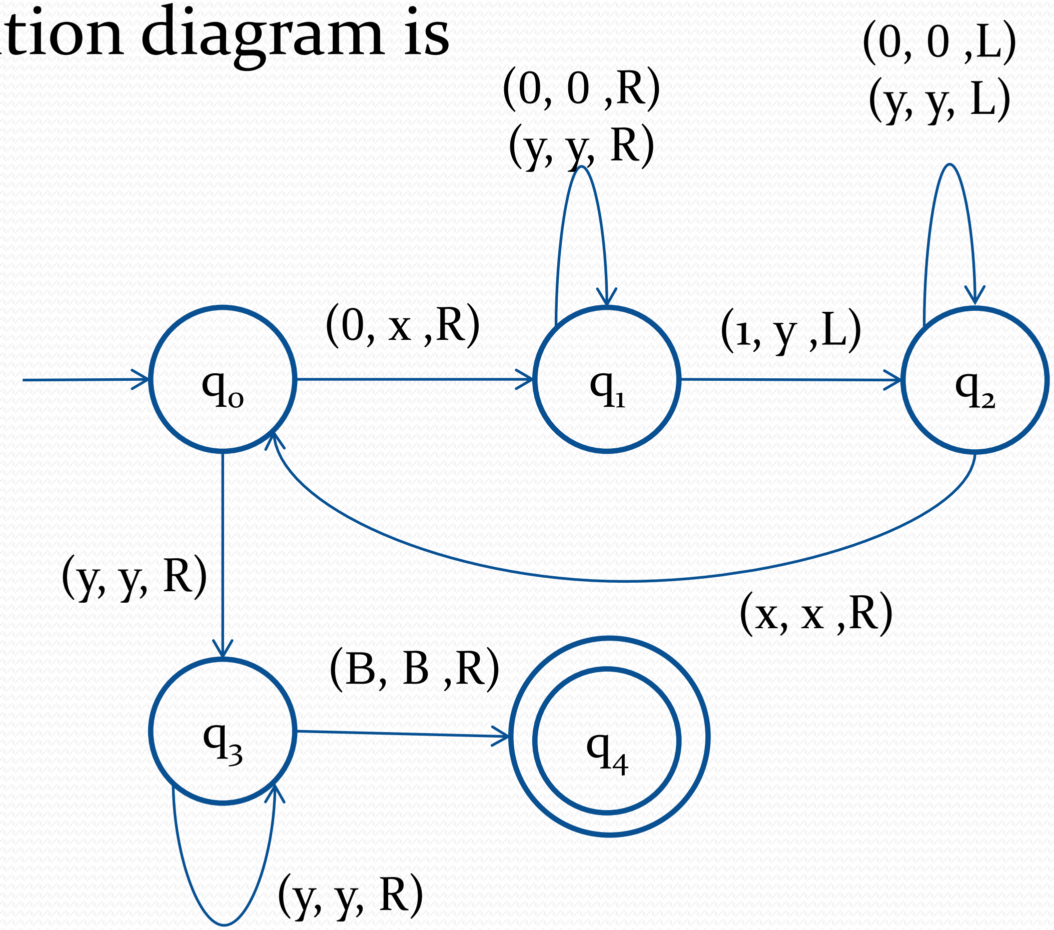
$L = \{ 0^n 1^n ! n \geq 1 \}$

Transition table is

δ	Tape symbols				
States	**0**	**1**	**x**	**y**	**B**
q_0	(q_1, x, R)			(q_3, y, R)	
q_1	$(q_1, 0, R)$	(q_2, y, L)		(q_1, y, R)	
q_2	$(q_2, 0, L)$		(q_0, x, R)	(q_2, y, L)	
q_3				(q_3, y, R)	(q_4, B, R)
q_4					

$L = \{ 0^n 1^n ! \ n \geq 1 \}$

Transition diagram is



Processing and Verification of TM

Acceptance

Consider string w = 0011 .

$q_0 0011 \vdash xq_1 011 \vdash x0q_1 11 \vdash xq_2 0y1 \vdash q_2 x0y1 \vdash xq_0 0y1 \vdash xxq_1 y1 \vdash xxyq_1 1 \vdash xxq_2 yy \vdash xq_2 xyy \vdash xxq_0 yy \vdash xxyq_3 y \vdash xxyyq_3 B \vdash xxyyBq_4 B$ (machine halts at final state)

Since machine halts at final state, therefore this string is accepted by TM.

Rejection

Consider string w = 011 .

$q_0 011 \vdash xq_1 11 \vdash q_2 xy1 \vdash xq_0 y1 \vdash xyq_3 1$ (machine halts at non-final state)

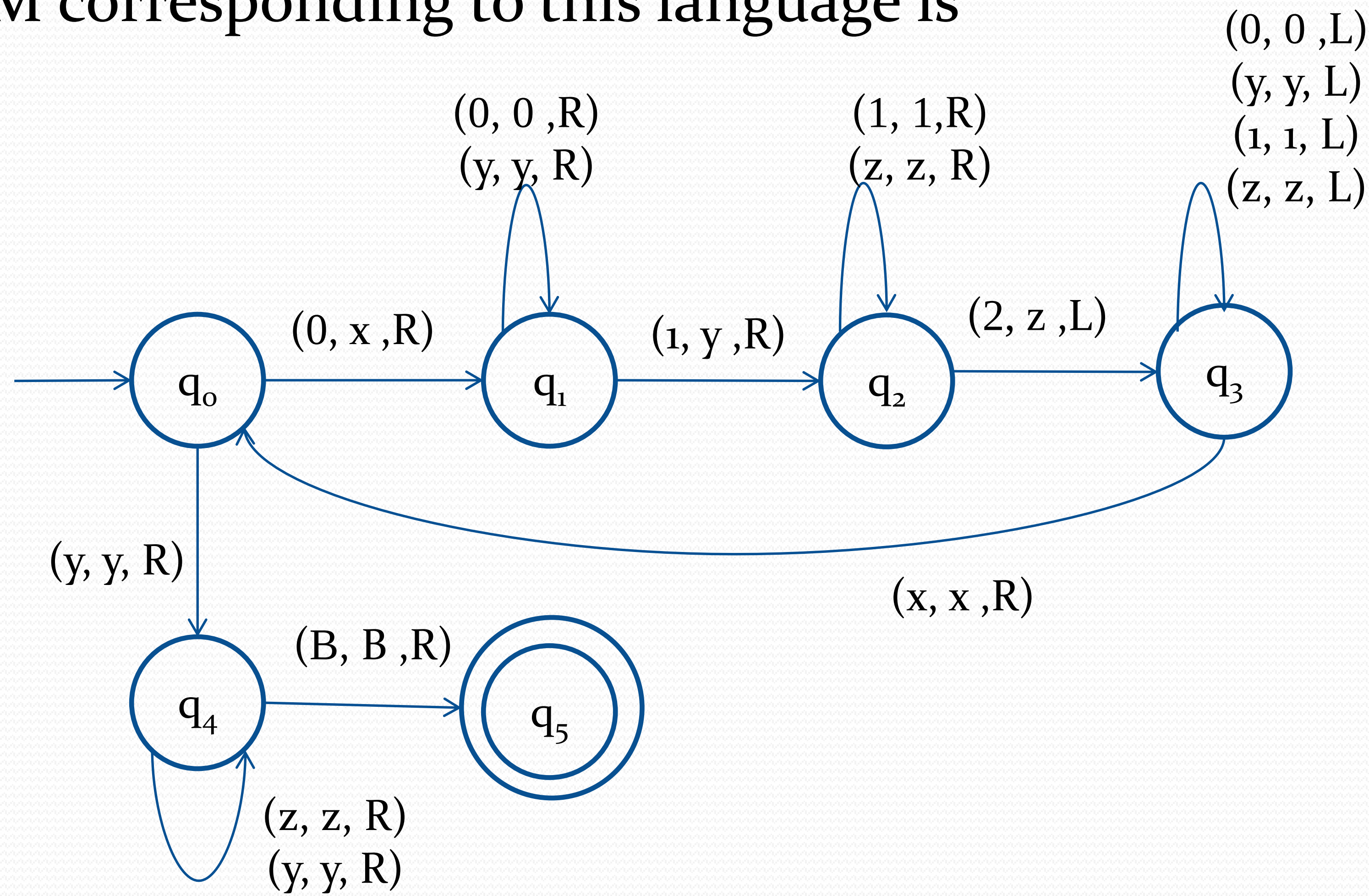
Since machine halts at non-final state, therefore this string is not accepted by TM.

Ex. Construct Turing machine for the language

$L = \{ 0^n 1^n 2^n ! \ n \geq 1 \}$.

Solution:

The TM corresponding to this language is



Processing and Verification of TM

Acceptance

Consider string w = 001122 .

$q_0001122 \vdash xq_101122 \vdash x0q_11122 \vdash x0yq_2122 \vdash x0y1q_222 \vdash$
$x0yq_31z2 \vdash x0q_3y1z2 \vdash xq_30y1z2 \vdash q_3x0y1z2 \vdash xq_00y1z2 \vdash$
$xxq_1y1z2 \vdash xxyq_11z2 \vdash xxyyq_2z2 \vdash xxyyzq_22 \vdash xxyyq_3zz \vdash$
$xxyq_3yzz \vdash xxq_3yyzz \vdash xq_3xyyzz \vdash xxq_0yyzz \vdash xxyq_4yzz \vdash$
$xxyyq_4zz \vdash xxyyzq_4z \vdash xxyyzzq_4B \vdash xxyyzzBq_5B$

(machine halts at final state)

Since machine halts at final state, therefore this string is accepted by TM.

Rejection

Consider string w = 00112 .

$q_0 00112 \vdash xq_1 0112 \vdash x0q_1 112 \vdash x0yq_2 12 \vdash x0y1q_2 2 \vdash x0yq_3 1z \vdash x0q_3 y1z \vdash xq_3 0yq_1 1z \vdash q_3 x0y1z \vdash xq_0 0y1z \vdash xxq_1 y1z \vdash xxyq_1 1z \vdash xxyyq_2 z \vdash xxyyzq_2 B$

(machine halts at non-final state)

Since machine halts at non-final state, therefore this string is not accepted by TM.

Turing Machine

Part-4

Ex. Construct TM to accept the language $L = \{ wcw^R ! w \in \{a, b\}^* \}$

Solution:

Some strings of this set are c, aca, bcb, abcba, bacab, aabcbaa etc.

Clearly all these strings are palindrome. That is, first symbol and last symbol are same. Similarly, second symbol and second last symbol are same , and so on.

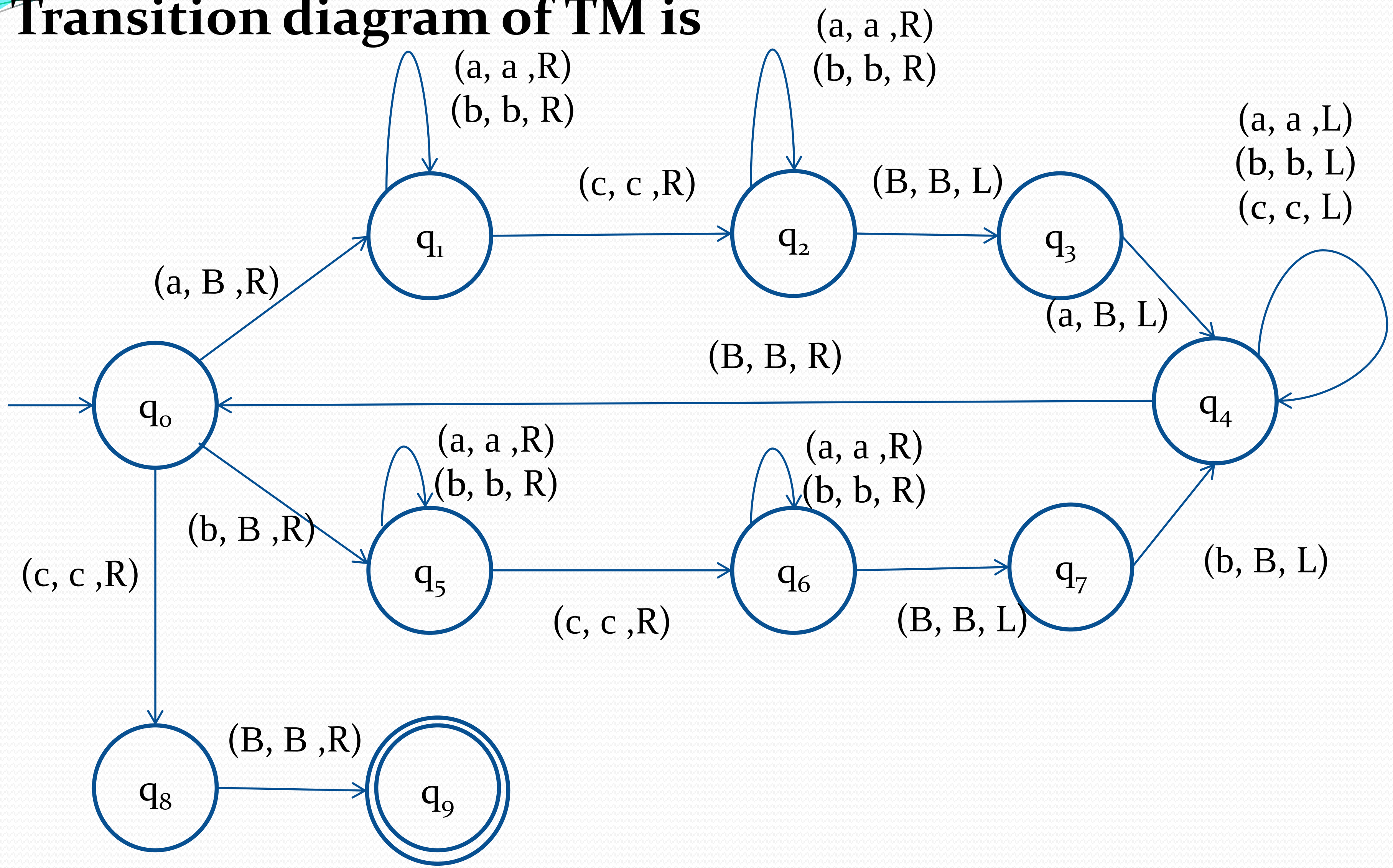
Procedure: TM is constructed in following steps. Let q_o is the initial state.

If the first input symbol is a, then remove it and change its state to q_1. After this, machine move to the last input symbol, if last input symbol is a, then machine remove it and back to first input symbol of string. This process continue.

If the first input symbol is b, then remove it and change its state to q_5. After this, machine move to the last input symbol, if last input symbol is b, then machine remove it and back to first input symbol of string. This process continue.

$L = \{ wcw^R \mid w \in \{a, b\}^* \}$

Transition diagram of TM is



Processing and Verification of TM

Acceptance

Consider string w = aabcbaa .

$q_0aabcbaa \vdash Bq_1abcbaa \vdash Baq_1bcbaa \vdash Babq_1cbaa \vdash Babcq_2baa \vdash Babcbq_2aa \vdash Babcbaq_2a \vdash Babcbaaq_2B \vdash Babcbaq_3aB \vdash Babcbq_4aB \vdash Babcq_4baB \vdash Babq_4cbaB \vdash Baq_4bcbaB \vdash Bq_4abcbaB \vdash q_4BabcbaB \vdash Bq_0abcbaB \vdash BBq_1bcbaB \vdash BBbq_1cbaB \vdash BBbcq_2baB \vdash BBbcbq_2aB \vdash BBbcbaq_2B \vdash BBbcbq_3aB \vdash BBbcq_4bBB \vdash BBbq_4cbBB \vdash BBq_4bcbBB \vdash Bq_4BbcbBB \vdash BBq_0bcbBB \vdash BBBq_5cbBB \vdash BBBcq_6bBB \vdash BBBcbq_6BB \vdash BBBcq_7bBB \vdash BBBq_4cBBB \vdash BBq_4BcBBB \vdash BBBq_0cBBB \vdash BBBcq_8BBB \vdash BBBcBq_9BB$

(machine halts at final state)

Since machine halts at final state, therefore this string is accepted by TM.

Rejection

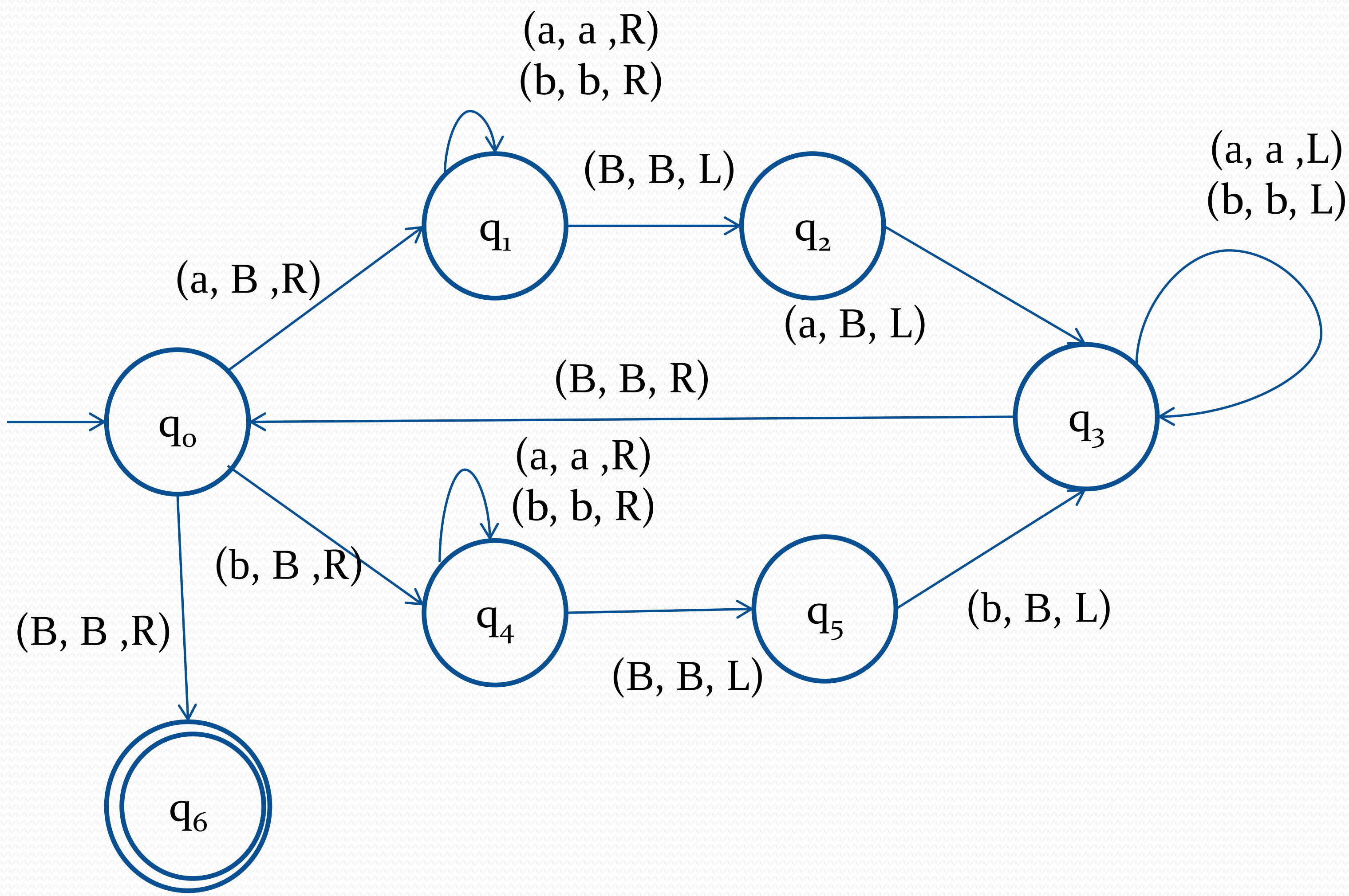
Consider string w = abcaa .

$q_0abcaa \vdash Bq_1bcaa \vdash Bq_1bcaa \vdash Bbq_1caa \vdash Bbcq_2aa \vdash Bbcaq_2a \vdash Bbcaaq_2B \vdash Bbcaq_3aB \vdash Bbcq_4aB \vdash Bbq_4caB \vdash Bq_4bcaB \vdash q_4BbcaB \vdash Bq_0bcaB \vdash BBq_5caB \vdash BBcq_6aB \vdash BBcaq_6B \vdash BBcq_7aB$ (machine halts at non-final state)

Since machine halts at non-final state, therefore this string is not accepted by TM.

Ex. Construct TM to accept the language $L = \{ ww^R \,!\, w \in \{a, b\}^* \}$

Transition diagram of TM is



Processing and Verification of TM

Acceptance

Consider string $w = abba.$

$q_0abba \vdash Bq_1bba \vdash Bbq_1ba \vdash Bbbq_1a \vdash Bbbaq_1B \vdash Bbbq_2aB \vdash Bbq_3bBB \vdash Bq_3bbBB \vdash q_3BbbBB \vdash Bq_0bbBB \vdash BBq_4bBB \vdash BBbq_4BB \vdash BBq_5bBB \vdash Bq_3BBBB \vdash BBq_0BBB \vdash BBBq_6BB$

(machine halts at final state)

Since machine halts at final state, therefore this string is accepted by TM.

Rejection

Consider string $w = abaa$.

$q_0abaa \vdash Bq_1baa \vdash Bbq_1aa \vdash Bbaq_1a \vdash Bbaaq_1B \vdash Bbaq_2aB \vdash Bbq_3aBB \vdash Bq_3baBB \vdash q_3BbaBB \vdash Bq_0baBB \vdash BBq_4aBB \vdash BBaq_4BB \vdash BBq_5aBB$ (machine halts at non-final state)

Since machine halts at non-final state, therefore this string is not accepted by TM.

Some additional problems

Construct TM for the following languages:-

(1) $L = \{ a^{n+2}b^n \text{ ! } n \geq 1\}$

(2) $L = \{ a^n b^n c^m \text{ ! } m, n \geq 0\}$

(3) $L = \{a^n b^n c^n \text{ ! } n \geq 1\}$

Turing Machine

Part-5

Turing computable function

Def. A function $f : N^n \rightarrow N$ is said to be Turing computable function if there exist a Turing machine which compute this function.

Here $N^n = N \times N \times N \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \times N$ (upto n times)

Note:

1) In the designing of Turing machine, we use unary number to represent a number. Here we use the unary number as a string of 1's.

Ex. 4 = 1111, 3 = 111 and so on.

2) If the function has multiple arguments, then we separate the arguments by 0.

Ex. Construct Turing machine for the following function

1) $f(n) = n+2$ $\qquad n \in N$

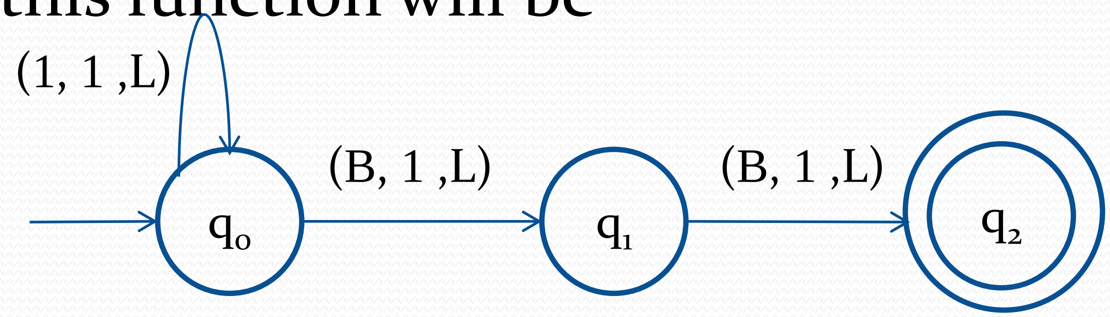
2) $f(m,n) = m+n$ $\qquad m, n \in N$

Solution:

1) In this function, if input is 1111 then output will be 111111.

i.e. $q_0 1111 \vdash^* q_f 111111$

TM for this function will be



Computation by this machine

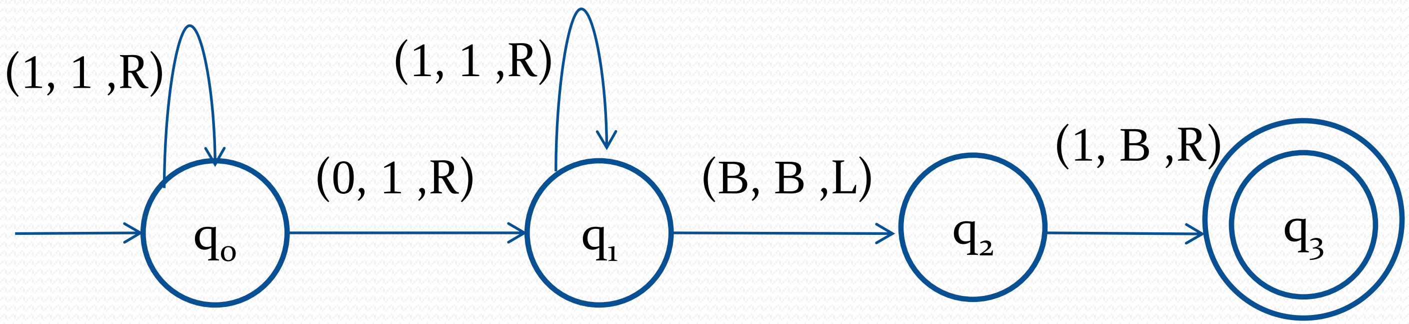
$q_0 1111 \vdash q_0 B1111 \vdash q_1 B11111 \vdash q_2 B111111$ (machine halt at final state)

(2) $f(m,n) = m+n \qquad m, n \in N$

Solution:

In this function if the input is 110111 then output will be 11111.

TM for this function will be



Computation by this machine

$q_0110111 \vdash 1q_010111 \vdash 11q_00111 \vdash 111q_1111 \vdash 1111q_111 \vdash 11111q_11 \vdash$
$111111q_1B \vdash 11111q_21\ B \vdash 11111Bq_3B$ (machine halt at final state)

Ex. Show that following function is Turing computable:-

$$f(n) = 3*n \qquad n \geq 1$$

Solution:

A function is said to be Turing computable if there exist a TM for this.

Therefore, we shall construct TM for this function.

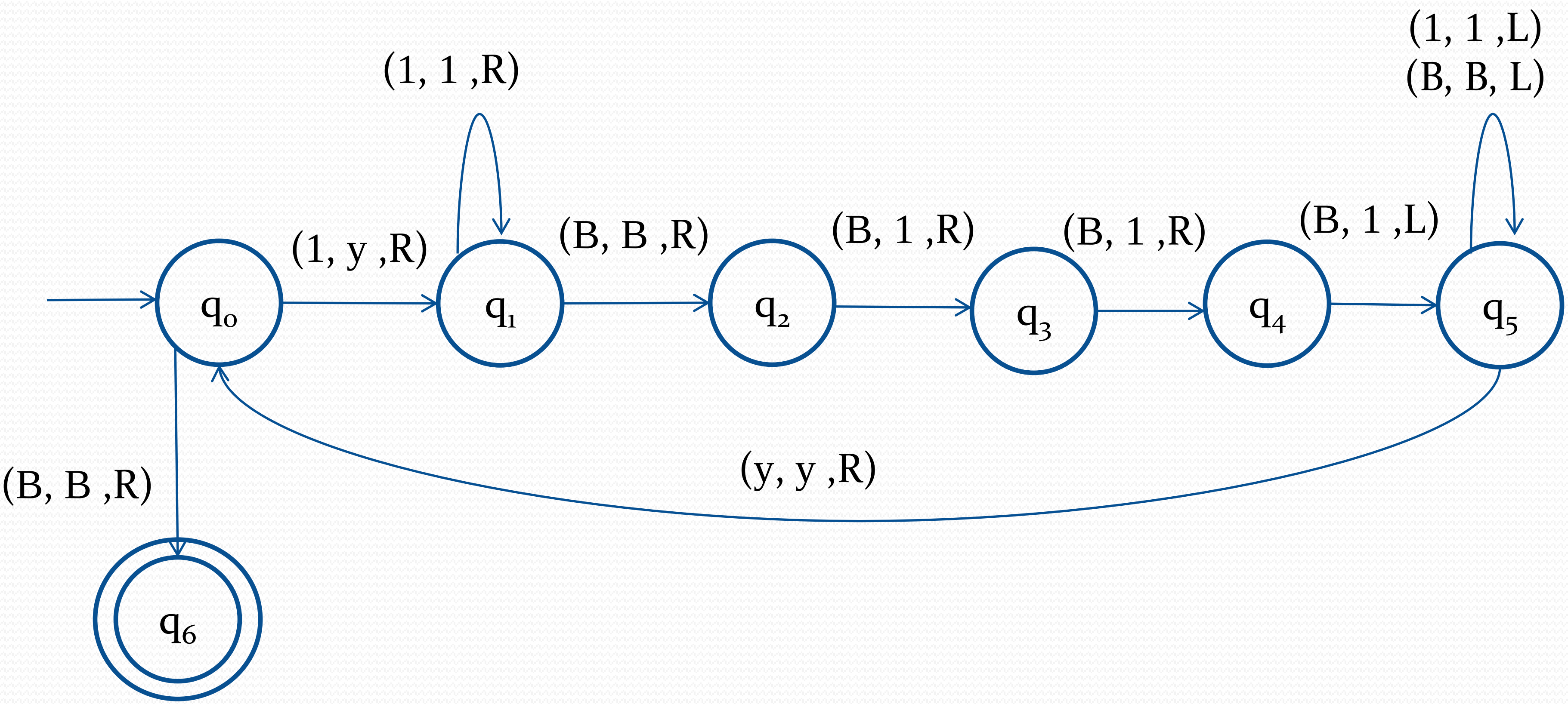
In this function, if the input is 2 then output will be 6. That is if input is 11 then output will be 111111.

First, we shall show that how string 11 is converted into 111111. After this, we construct Turing machine for this.

Suppose initial state is q_o.

$q_o11 \vdash yq_11 \vdash y1q_1B \vdash y1Bq_2B \vdash y1B1q_3B \vdash y1B11q_4B \vdash y1B1q_511$
$\vdash y1Bq_5111 \vdash y1q_5B111 \vdash yq_51B111 \vdash q_5y1B111 \vdash yq_01B111$
$\vdash yyq_1B111 \vdash yyBq_2111 \vdash yyB1q_211 \vdash yyB11q_21 \vdash yyB111q_2B$
$\vdash yyB1111q_3B \vdash yyB11111q_4B \vdash yyB1111q_511B \vdash yyB111q_5111B$
$\vdash yyB11q_51111B \vdash yyB1q_511111B \vdash yyBq_5111111B$
$\vdash yyq_5B111111B \vdash yq_5yB111111B \vdash yyq_0B111111B \vdash yyBq_6111111B$

Therefore, the Turing machine corresponding above function is constructed as following:-



Turing Machine

Part-6

Ex. Show that following function is Turing computable:-

$$f(m, n) = m-n \qquad m \geq n$$
$$= 0 \qquad \text{otherwise}$$

Solution: Clearly this function is proper subtraction function.

We have to find TM corresponding to this function.

There are two cases of this function.

Case-1: If $m \geq n$ then value of the function is m-n. i.e. if m= 6 and n=4 then value = 2.

Case-2: If $m < n$ then value of the function is 0. i.e. if m= 4 and n=6 then value = 0.

Before constructing TM for this function, first we process the input and develop rules through which machine move from initial ID to final ID.

Case-1: when m ≥ n.

$q_0 1111011 \vdash 1q_0 111011 \vdash 11q_0 11011 \vdash 111q_0 1011 \vdash 1111q_0 011 \vdash 11110q_1 11 \vdash 1111q_2 0y1 \vdash 111q_2 10y1 \vdash 111yq_0 0y1 \vdash 111y0q_1 y1 \vdash 111y0yq_1 1 \vdash 111y0q_2 yy \vdash 111yq_2 0yy \vdash 111q_2 y0yy \vdash 11q_2 1y0yy \vdash 11yq_0 y0yy \vdash 11yyq_0 0yy$

$\vdash 11yy0q_1 yy \vdash 11yy0yq_1 y \vdash 11yy0yyq_1 B \vdash 11yy0yq_3 yB$

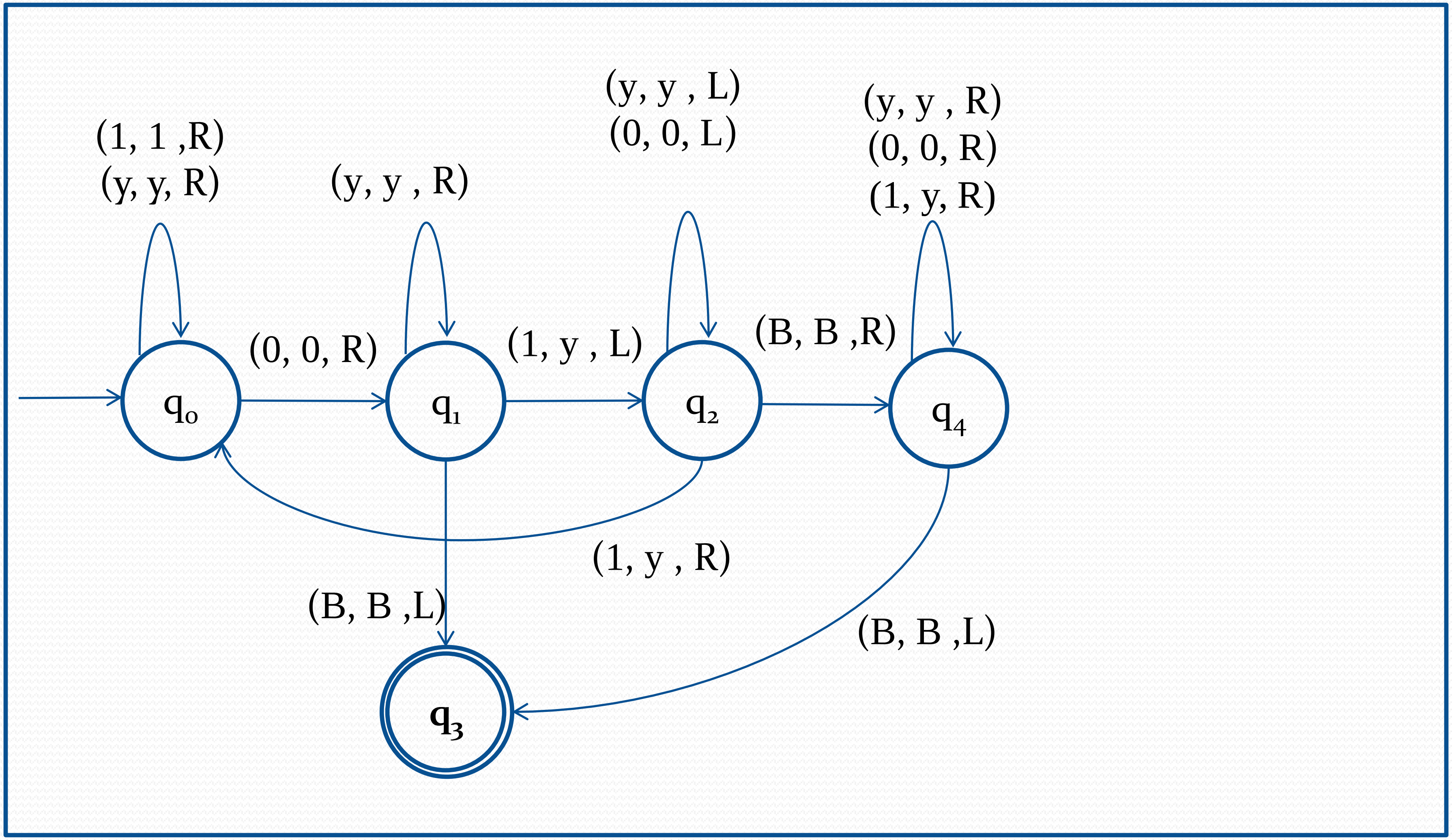
(machine halts at final state)

Case-2: when m < n.

$q_0 1101111 \vdash^* yy0yyq_1 11 \vdash yy0yq_2 yy1 \vdash^* q_2 Byy0yyy1 \vdash Bq_4 yy0yyy1 \vdash^* Byy0yyyq_4 1 \vdash Byy0yyyyq_4 B \vdash Byy0yyyq_3 yB$

(machine halts at final state)

Therefore, the TM corresponding this function will be constructed as following:-



Ex. Construct Turing machine for the following function

f(m,n) = m*n   m, n ∈ N

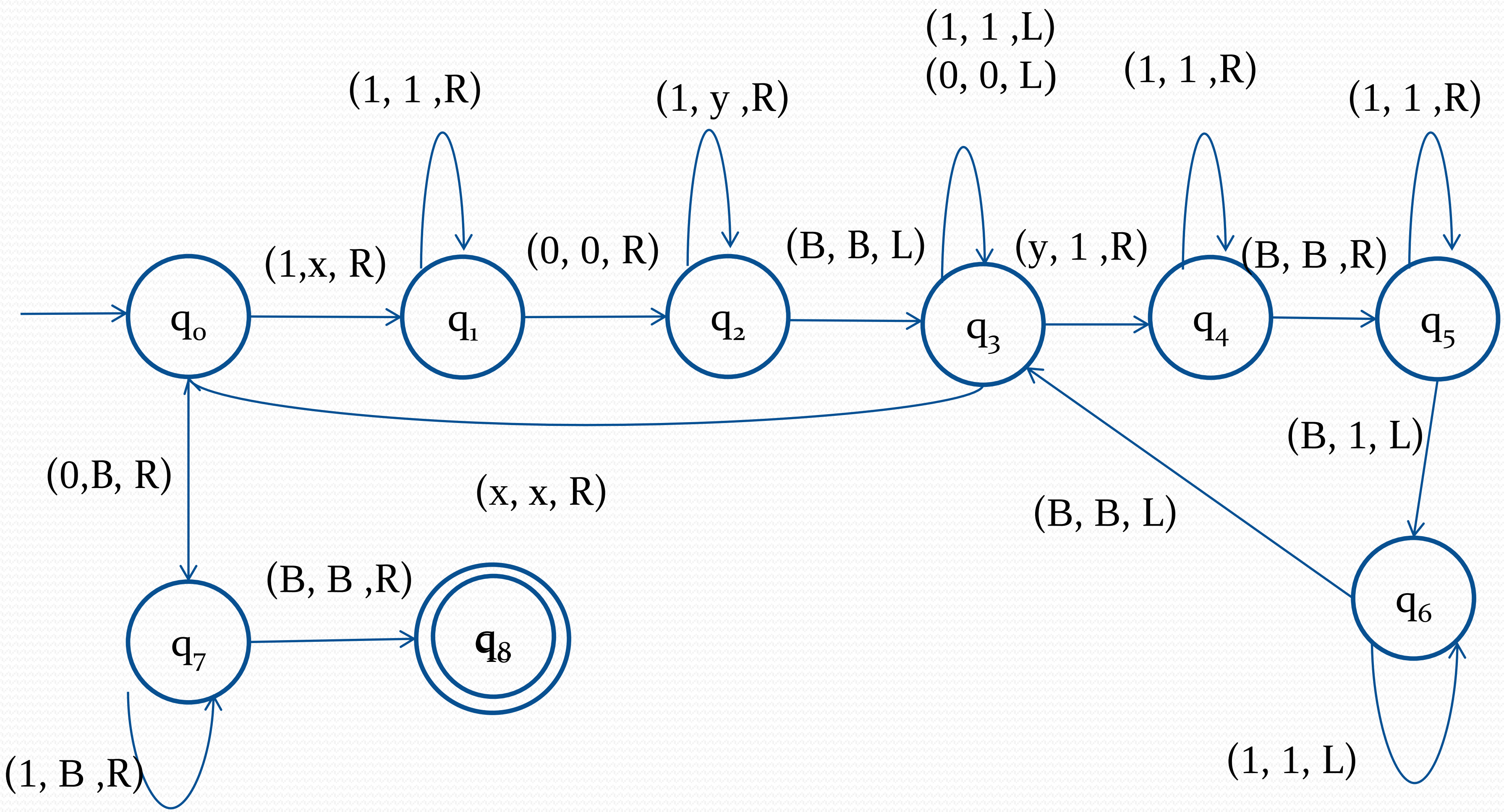
Solution: This function multiply two numbers.
If inputs are 2 and 3 then output will be 6.
Processing:

$q_0110111 \vdash xq_110111 \vdash x1q_10111 \vdash x10q_2111 \vdash x10yq_211 \vdash$

$x10yyq_21 \vdash x10yyyq_2B \vdash x10yyq_3yB \vdash x10yy1q_4B \vdash x10yy1B\,q_5B \vdash$
$x10yy1q_6B1 \vdash x10yyq_31B1 \vdash x10yq_3y1B1$

$\vdash x10y1q_41B1 \vdash x10y11q_4B1 \vdash x10y11Bq_51 \vdash x10y11B1q_5B \vdash$
$x10y11Bq_611 \vdash x10y11q_6B11 \vdash x10y1q_31B11 \vdash x10yq_311B11 \vdash$
$x10q_3y11B11 \vdash x101q_411B11 \vdash x1011q_41B11 \vdash x10111q_4B11 \vdash$
$x10111Bq_511 \vdash x10111B1q_51 \vdash x10111B11q_5B \vdash x10111B1q_611 \vdash$
$x10111Bq_6111 \vdash x10111q_6B111 \vdash x1011q_31B111 \vdash x101q_311B111 \vdash$
$x10q_3111B111 \vdash x1q_30111B111 \vdash xq_310111B111 \vdash q_3x10111B111 \vdash$
$xq_010111B111 \vdash {}^*xxq_00111B111111 \vdash xxBq_7111B111111$
$\vdash xxBBq_711B111111 \vdash xxBBBq_71B111111 \vdash xxBBBBq_7B111111$
$\vdash xxBBBBBq_8111111$ (machine halts at final state)

Therefore, the TM corresponding this function will be constructed as following:-



Turing Machine

Part-7

Variations or types of TM

1) Non-deterministic Turing Machine(TM)
2) Multi-tape Turing Machine(TM)
3) Multi-head Turing Machine(TM)
4) Multi-directional Turing Machine(TM)

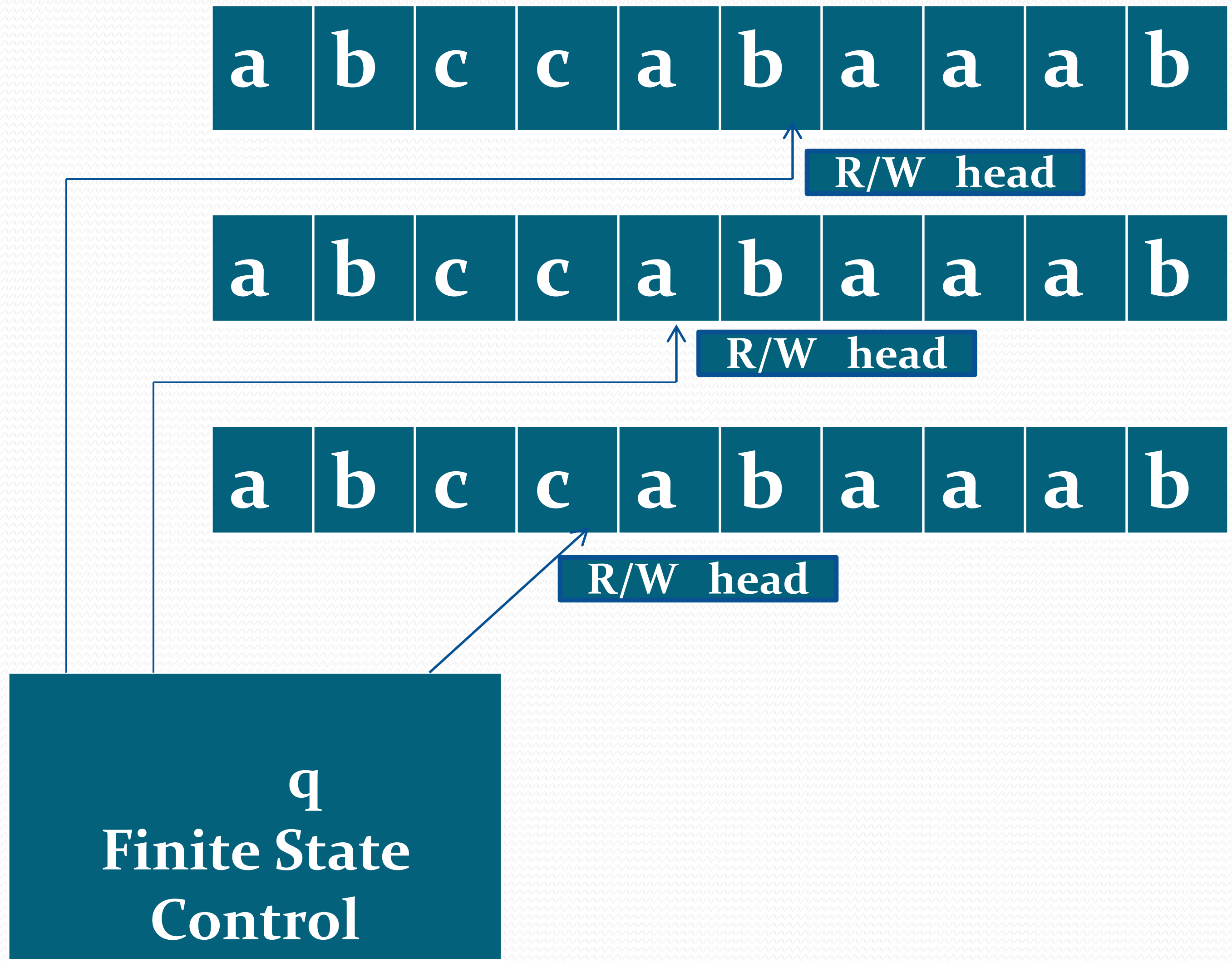
Non-deterministic Turing Machine(TM)

- A non-deterministic TM is a Turing machine which, like nondeterministic finite automata, at any state it is in and for the tape symbol it is reading, can take any action selecting from a set of specified actions rather than taking one definite predetermined action.
- Even in the same situation it may take different actions at different times.
- It differs from deterministic TM only by transition function.
- The transition function of non-deterministic TM is defined as following:-

$$\delta: Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L, R\}}$$

Multi-tape Turing Machine(TM)

Model of TM



Multi-tape Turing Machine(TM)

This type of machine consists of n number of tapes. Since number of tapes is n, therefore the number of heads will also be n.

Transition function will be

$$\delta: Q\times\Gamma^n \rightarrow Q\times\Gamma^n\times\{L, R\}^n$$

Where

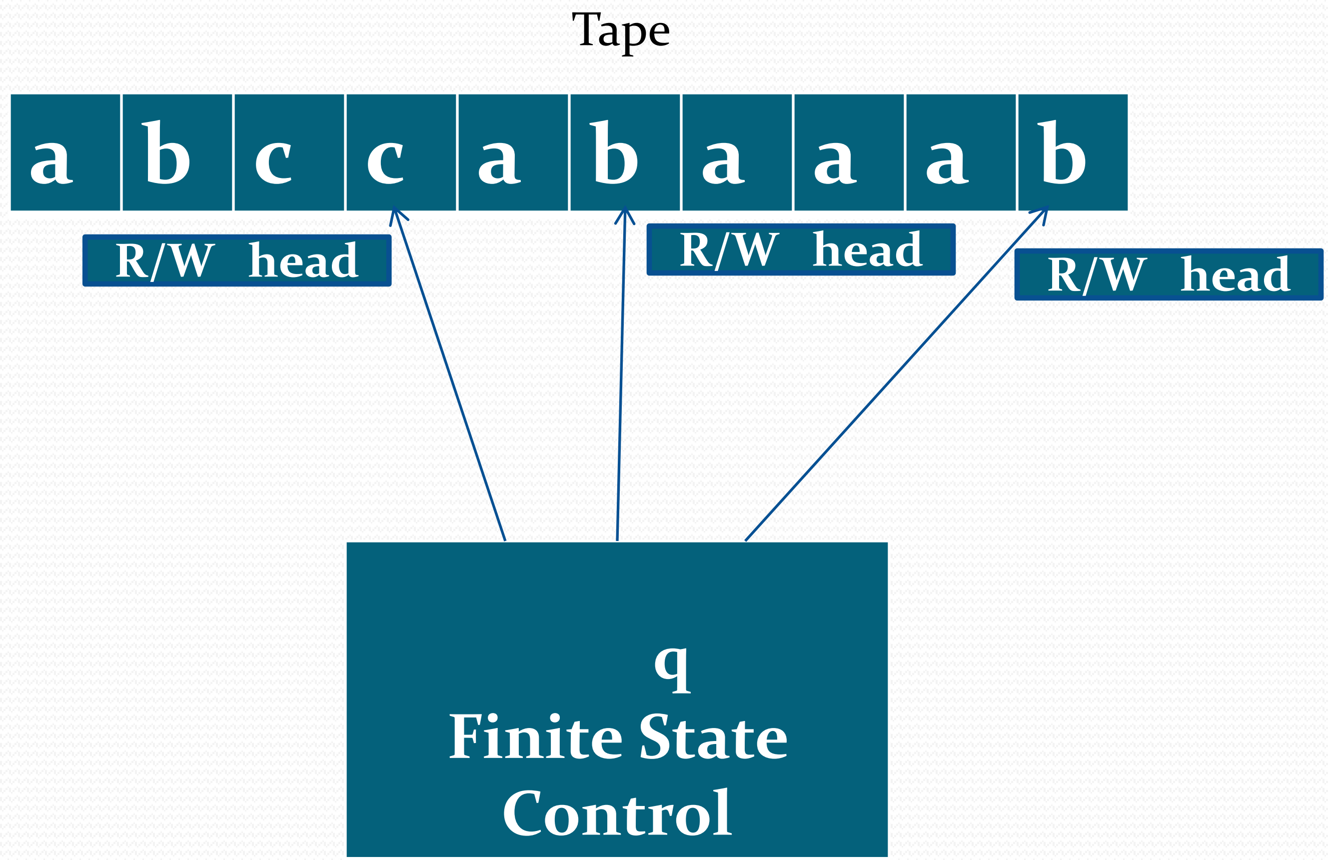
$$\Gamma^n = \Gamma\times \Gamma\times \Gamma\times\ldots\ldots\ldots\ldots\ldots\ldots\ldots \times \Gamma(\text{upto n times})$$

$$\{L, R\}^n = \{L, R\}\times \{L, R\}\times \{L, R\}\times\ldots\ldots\ldots\ldots\ldots\ldots \times \{L, R\}$$

(upto n times)

Multi-head Turing Machine(TM)

Model of TM



Multi-head Turing Machine(TM)

This type of machine consists of one tape with n heads.

Transition function will be

$$\delta: Q \times \Gamma^n \rightarrow Q \times \Gamma^n \times \{L, R\}^n$$

Where

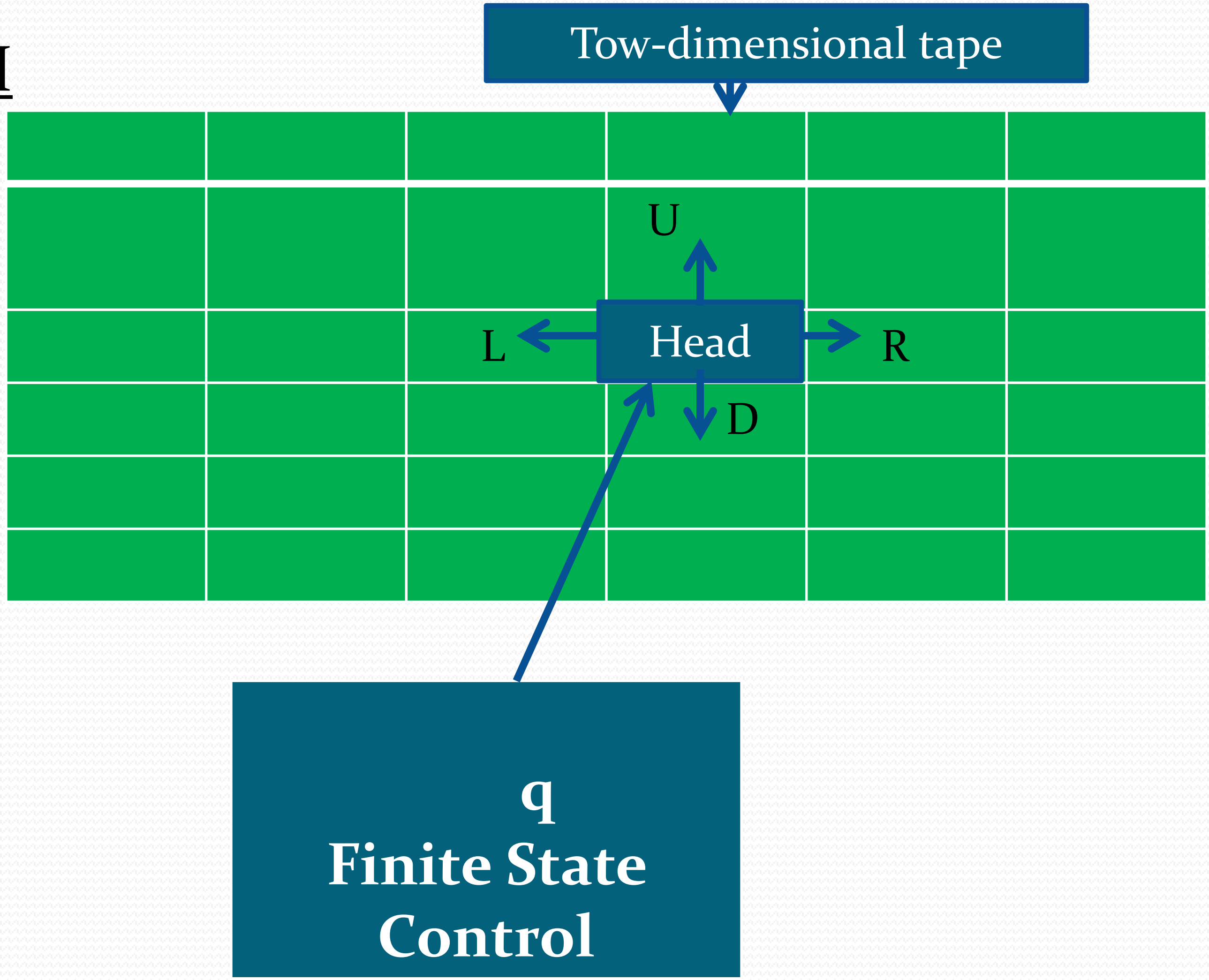
$$\Gamma^n = \Gamma \times \Gamma \times \Gamma \times \ldots\ldots\ldots\ldots\ldots\ldots \times \Gamma \text{(upto n times)}$$

$$\{L, R\}^n = \{L, R\} \times \{L, R\} \times \{L, R\} \times \ldots\ldots\ldots\ldots\ldots\ldots \times \{L, R\}$$

(upto n times)

Multi-dimensional Turing Machine(TM)

Model of TM



Multi-dimensional Turing Machine(TM)

- This type of machine consists of one multi-dimensional tape with one heads.
- The head of machine move in many directions.
- If tape is n-dimensional then head move in 2^n directions.
- Transition function of two-dimensional TM is defined as

$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, U, D\}$

Where

L→ Left direction

R→ Right direction

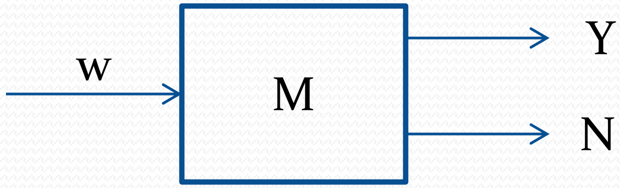
U→ Up direction

D→ Down direction

Recursive and Recursive Enumerable language

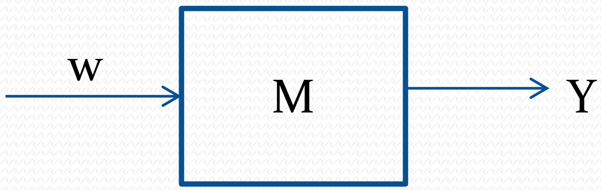
Recursive language:

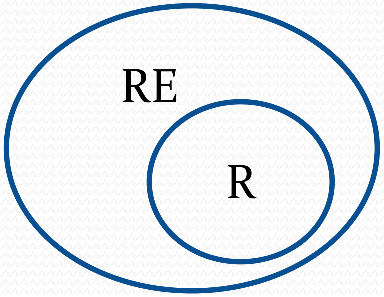
A language L is said to be recursive if there exists a Turing machine M which accepts all the strings w belong into L and rejects all the strings which do not belong into L.



Recursive Enumerable language:

A language L is said to be recursive enumerable if there exists a Turing machine M which accepts all the strings w belong into L and rejects or goes into an infinite loop for all the strings which do not belong into L.



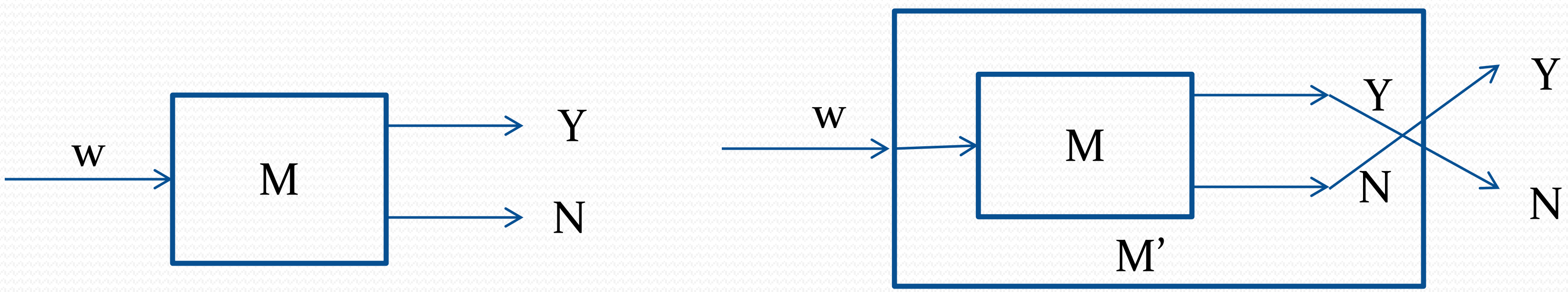


Properties of Recursive and Recursive Enumerable languages

1) If L is recursive language then $\bar{L}$ is also recursive language.
2) If L and $\bar{L}$ are recursive enumerable languages then L will be recursive language.
3) The union of two recursive languages is also recursive i.e. if L_1 and L_2 are recursive then $L_1 \cup L_2$ will be also recursive.
4) The union of two recursive enumerable languages is also recursive enumerable i.e. if L_1 and L_2 are recursive enumerable then $L_1 \cup L_2$ will be also recursive enumerable.
5) The intersection of two recursive languages is also recursive i.e. if L_1 and L_2 are recursive then $L_1 \cap L_2$ will be also recursive.
6) The intersection of two recursive enumerable languages is also recursive enumerable i.e. if L_1 and L_2 are recursive enumerable then $L_1 \cap L_2$ will be also recursive enumerable .

Theorem: If L is recursive language then complement of L i.e. $\bar{L}$ is also recursive language.

Proof: Since L is recursive language, therefore there exists a TM which accepts all strings belong into L and rejects all strings which do not belong into L. Let this TM is M. Therefore M will be



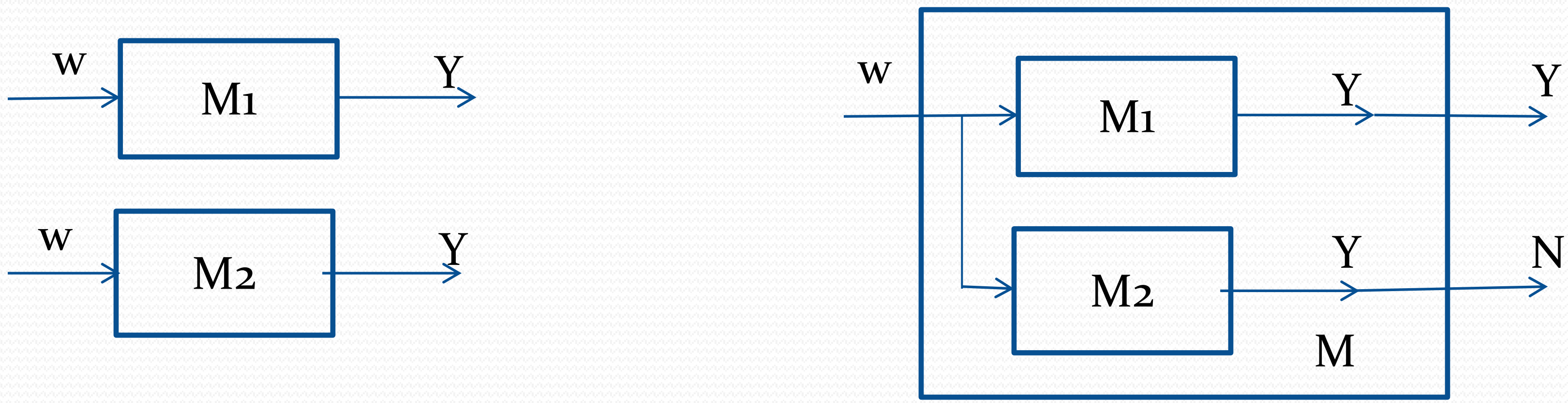
Now, we construct a TM M’ using M as above.

Clearly, if w is accepted by M then w is rejected by M’ and if w is rejected by M then w is accepted by M’. Since L is accepted by M, therefore complement of L i.e. $\bar{L}$ is accepted by M’.

Since there exists a TM M’ corresponding to $\bar{L}$, therefore $\bar{L}$ is recursive language.

Theorem: If L and $\bar{L}$ are recursive enumerable languages then L will be recursive language.

Proof: Since L and $\bar{L}$ are recursive enumerable language, therefore there exists TM M1 and M2 corresponding to L and $\bar{L}$ respectively. M1 accepts all strings belong into L and M2 accepts all strings belong into and $\bar{L}$. These are



Now, we construct a TM M using M1 and M2 as above.

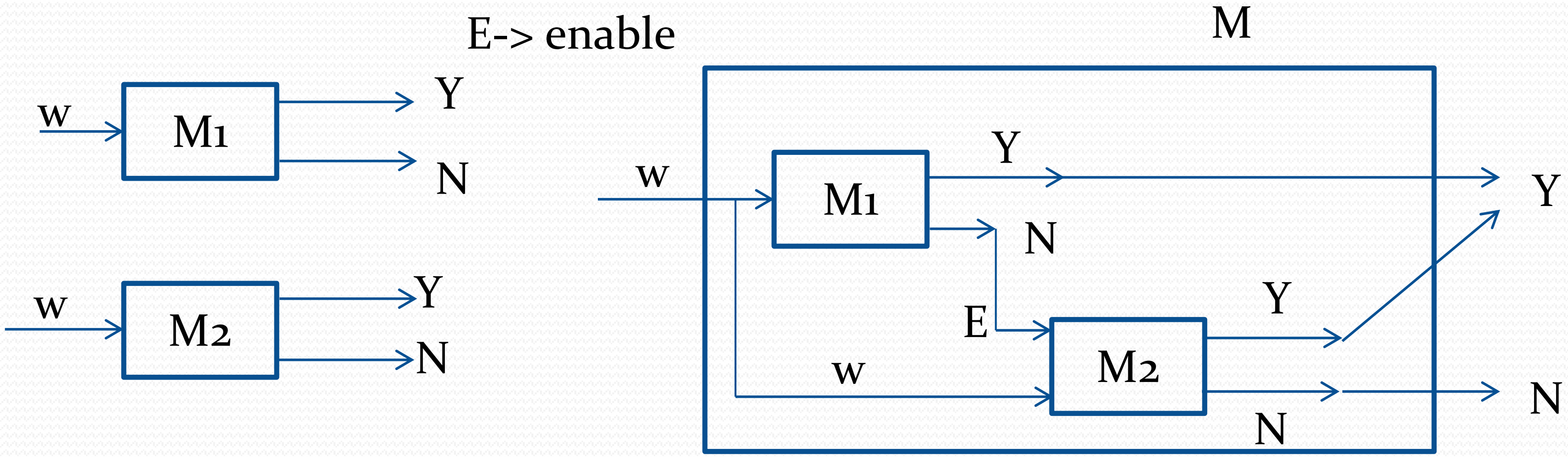
Clearly, if w is accepted by M1 then w is also accepted by M and if w is accepted by M2 then w is rejected by M. That is, if $w \in L$ then it is accepted by M and if $w \notin L$ then it is rejected by M.

Therefore, M is a TM which accepts all strings of L and rejects all strings which are not belong into L.

Hence L is recursive language.

Theorem: The union of two recursive languages is also recursive i.e. if L1 and L2 are recursive then L1∪ L2 will be also recursive.

Proof: Since L1 and L2 are recursive languages then there exists TM M1 and M2 corresponding to L1 and L2 respectively are of the form:



Consider a string w ∈ L1∪ L2. Then w ∈ L_1 or w ∈ L_2.

If w ∈ L_1 then it is accepted by M_1. therefore it is also accepted by M. If w ∈ L_2 then it is accepted by M_2. therefore it is also accepted by M.

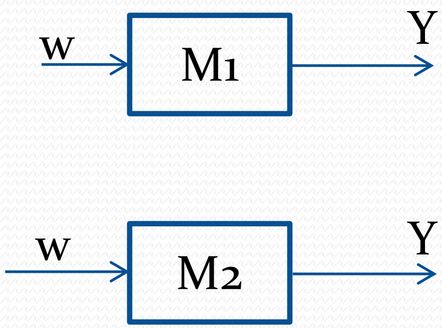
But if w ∉ L1∪ L2 then w is neither accepted by M_1 nor M_2. therefore it is also not accepted by M.

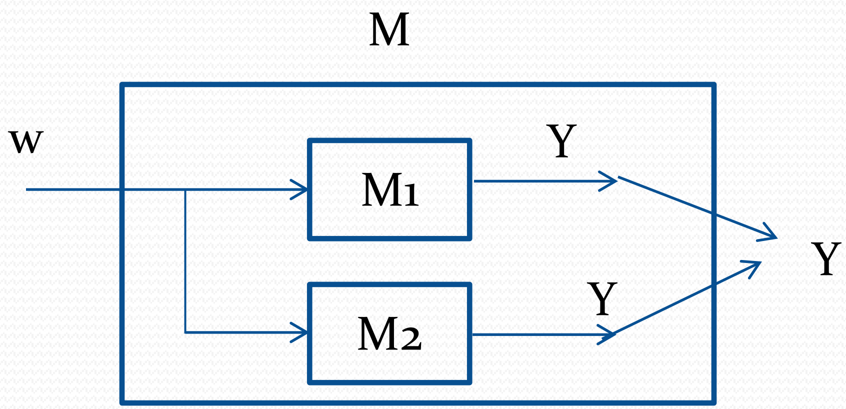
Hence M is a machine which accepts all strings belong into L1∪ L2 and rejects all strings which do not belong into L1∪ L2.

Therefore L1∪ L2 is recursive language.

Theorem: The union of two recursive enumerable languages is also recursive enumerable i.e. if L_1 and L_2 are recursive enumerable then $L_1 \cup L_2$ will be also recursive enumerable.

Proof: Since L_1 and L_2 are recursive enumerable languages then there exists TM M_1 and M_2 corresponding to L_1 and L_2 respectively are of the form:





Consider a string $w \in L_1 \cup L_2$. Then $w \in L_1$ or $w \in L_2$.

If $w \in L_1$ then it is accepted by M_1. therefore it is also accepted by M. If $w \in L_2$ then it is accepted by M_2. therefore it is also accepted by M.

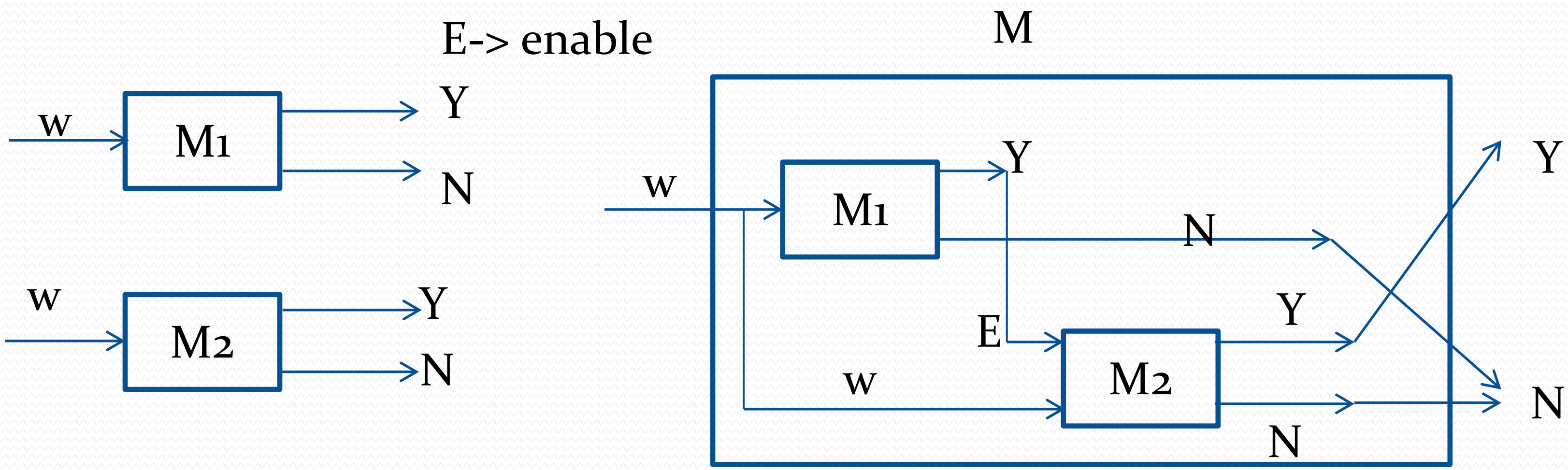
But if $w \notin L_1 \cup L_2$ then w is neither accepted by M_1 nor M_2. therefore it is also not accepted by M.

Hence M is a machine which accepts all strings belong into $L_1 \cup L_2$ and rejects all strings which do not belong into $L_1 \cup L_2$.

Therefore $L_1 \cup L_2$ is recursive language.

Theorem: The intersection of two recursive languages is also recursive i.e. if L1 and L2 are recursive then $L1 \cap L_2$ will be also recursive.

Proof: Since L1 and L2 are recursive languages then there exists TM M1 and M2 corresponding to L1 and L2 respectively are of the form:



Consider a string $w \in L1 \cap L_2$. Then $w \in L_1$ and $w \in L_2$.

Since $w \in L_1$ therefore it is accepted by M_1. therefore it is also accepted by M. Since w $\in L_2$ therefore it is accepted by M_2. Clearly, therefore it is also accepted by M.

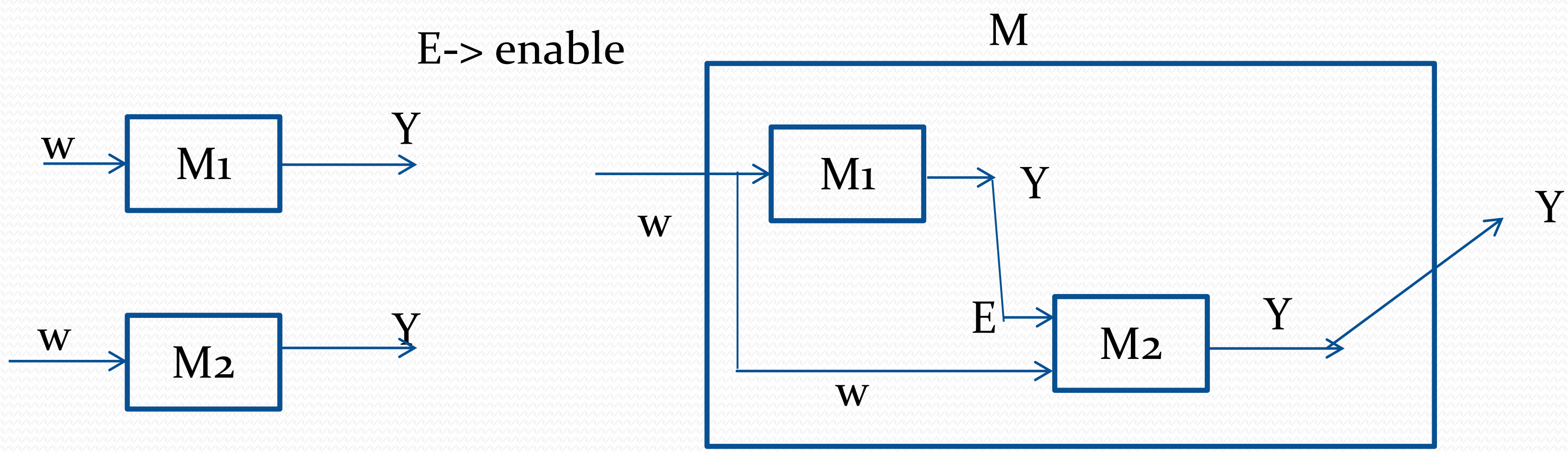
But if $w \notin L1 \cap L_2$ then w is either not belong into L1 or not belong into L_2. therefore it is either not accepted by M_1 or not accepted by M_2. Clearly, therefore w is not accepted by M.

Hence M is a machine which accepts all strings belong into $L1 \cap L_2$ and rejects all strings which do not belong into $L1 \cap L_2$.

Therefore $L1 \cap L_2$ is recursive language.

Theorem: The intersection of two recursive enumerable languages is also recursive enumerable i.e. if L1 and L2 are recursive enumerable then L1$\cap$ L_2 will be also recursive enumerable.

Proof: Since L1 and L2 are recursive enumerable languages then there exists TM M1 and M2 corresponding to L1 and L2 respectively are of the form:



Consider a string w $\in$ L1$\cap$ L_2. Then w $\in$ L_1 and w $\in$ L_2.

Since w $\in$ L_1 and w $\in$ L_2 , therefore it is accepted by both M_1 and M_2. Clearly, therefore it is also accepted by M.

But if w $\notin$ L1$\cap$ L_2 then w is either not belong into L_1 or not belong into L_2. In this case, we can not say that w is accepted or not accepted by M_1 or M_2. Clearly, therefore we can also say that w is accepted or not by M.

Hence M is a machine which accepts all strings belong into L1$\cap$ L_2 and rejects or goes into infinite loop for all strings which do not belong into L1$\cap$ L_2 .

Therefore L1$\cap$ L_2 is recursive enumerable language.

Post Correspondence Problem(PCP)

The PCP problem over an alphabet ∑ is stated as follows:–

Given the following two lists, **X** and **Y** of non-empty strings over ∑,

$$X = (x_1, x_2, x_3, \ldots\ldots\ldots, x_n)$$

$$Y = (y_1, y_2, y_3, \ldots\ldots\ldots, y_n)$$

We can say that there is a Post Correspondence Solution, if for some $(i_1, i_2, \ldots\ldots\ldots\ldots i_k)$, where $1 \le i_j \le n$, the condition

$$x_{i1}\, x_{i2} \ldots\ldots x_{ik} = y_{i1}\, y_{i2} \ldots\ldots y_{ik}$$ satisfies.

Ex. Find whether the lists X = (abb, aa, aaa) and Y = (bba, aaa, aa) have a Post Correspondence Solution?

Solution:

Here,

$$\mathbf{x_2x_1x_3} = \text{'}\mathbf{aaabbaaa}\text{'}$$

and $\mathbf{y_2y_1y_3} = \text{'}\mathbf{aaabbaaa}\text{'}$

We can see that

$$\mathbf{x_2x_1x_3 = y_2y_1y_3}$$

Hence, the solution is **(2, 1,3).** Another solution may be also **(2, 3), (3, 2).**

Ex. Find whether the lists X = (b, bab^3 ,ba) and
Y = (b^3, ba, a) have a Post Correspondence Solution?

Solution:

$$\mathbf{x_2x_1x_1x_3 = bab^3\ b\ b\ ba}$$

and $\mathbf{y_2y_1y_1y_3 = ba\ b^3\ b^3\ a}$

Therefore the solution will be (2, 1, 1, 3).

Ex. Find whether the lists X = (ab, bab, bbaaa)and
Y = (a, ba, bab) have a Post Correspondence Solution?

Solution:

In this case there is no solution of this problem. Because the length of each string in Y is less than corresponding string in X.

That is $|y_i| < |x_i|, \ \forall\, i.$

Modified Post correspondence problem (MPCP)

The modified PCP problem over an alphabet ∑ is stated as follows:–

Given the following two lists, **X** and **Y** of non-empty strings over ∑,

$$X = (x_1, x_2, x_3, \ldots\ldots\ldots, x_n)$$

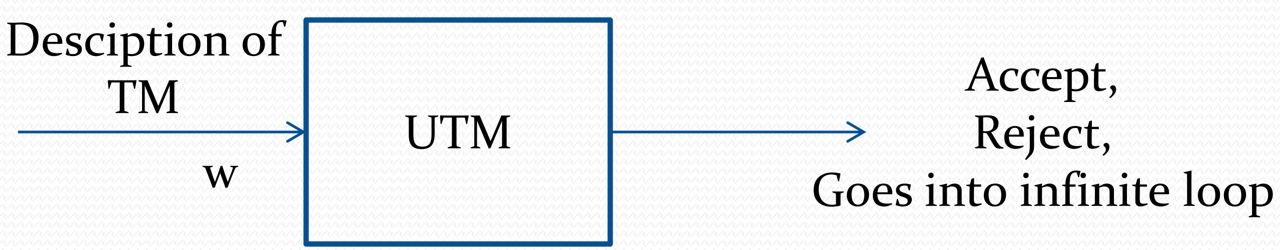
$$Y = (y_1, y_2, y_3, \ldots\ldots\ldots, y_n)$$

We can say that there is a Modified Post Correspondence Solution, if for some $(i_1, i_2, \ldots\ldots\ldots\ldots i_k)$, where $1 \leq i_j \leq n$, the condition $x_1 x_{i1} x_{i2} \ldots\ldots\ldots x_{ik} = y_1 y_{i1} y_{i2} \ldots\ldots\ldots y_{ik}$ satisfies.

Universal Turing machine(UTM)

- A universal Turing machine (UTM) behaves like a general purpose computer. Instead of finite size memory in computer, UTM uses infinite tape.
- UTM is a specified TM that can simulate the behavior any TM.
- UTM is a Turing machine that accepts universal language.
- Universal language is defined as:-

UL= { <M,w> ! M is a Turing machine that accepts input string w.}



Universal Turing machine(UTM)

Input to UTM:

Description of TM

Input string

Action of UTM:

Simulate TM

Behave like TM

UTM as Computer

TM as Program

UTM is a recognizer but not a decider.

UTM takes an encoding of a TM and the input data as its input in its tape and behaves as that TM on the input data.

Church-Turing Thesis

- It states that if there exists an algorithm to solve a problem then there exists a Turing machine to solve that problem and vice-versa.
- It states that a function on the natural numbers can be computed by an algorithm if and only if it is computable by a Turing machine.
- A problem can be solved by an algorithm iff it can be solved by a Turing Machine.
- Algorithm $\Longleftrightarrow$ Turing machine

Halting Problem

Statement: Given Turing machine M and input string w, **is it possible to determine whether the machine will ever halt on given input string?**

In another words, the **halting problem** is the problem of determining, from a description of an arbitrary computer program and an input, whether the program will finish running, or continue to run forever.

Halt: the machine will stop or halt at final or non-final state after finite number steps.

No halt: Machine will never stop or halt.

Decidable or Undecidable Problem

- A problem is said to be decidable if there exists an algorithm which can decide the problem in finite amount of time.
- In this type of problems, the output of the algorithm will be yes/no i.e. the answer of decidable problems is yes or no.
- A problem is said to be undecidable if there does not exist an algorithm which can decide the problem in finite amount of time.

Turing decidable and Turing acceptable language

- A language L is said to be Turing decidable if there exists a Turing machine which can accepts all strings belong in to L and rejects all strings which do not belong into L.
- A language L is said to be Turing acceptable if there exists a Turing machine which can accepts all strings belong in to L.

Some undecidable problems

- Halting problem is undecidabe.
- PCP problem is undecidable.
- Modified PCP problem is undecidable.
- For a CFG G, is L(G) ambiguous ?
- For two arbitrary CFG G_1 and G_2,
  deciding $L(G_1) \cap L(G_2) = \phi$ or not, is undecidable.

Some decidable problems

- For a CFG G, is L(G) = ϕ or not, is decidable.
- For a CFG G, finding whether L(G) a finite or not, is decidable.
- For regular language L_1 and L_2, finding whether $L_1 \cup L_2$ is regular, is decidable.
- Membership problem in CFG is decidable.