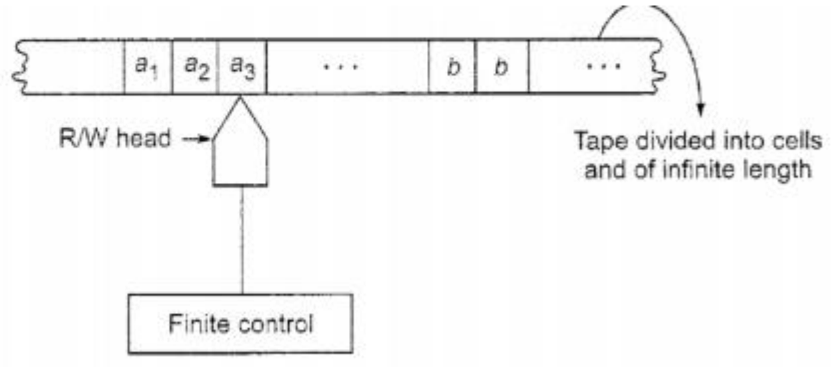
Module 5

Turing Machine Model

The Turing machine can be thought of as finite control connected to a R/W (read/write) head. It has one tape which is divided into a number of cells.



Turing machine model.

Each cell can store only one symbol. The input to and the output from the finite state automaton are affected by the R/W head which can examine one cell at a time.In one move, the machine examines the present symbol under the R/W head on the tape and the present state of an automaton to determine

(i) a new symbol to be written on the tape in the cell under the R/W head,

(ii) a motion of the R/W head along the tape: either the head moves one cell left (L). or one cell right (R),

(iii) the next state of the automaton, and

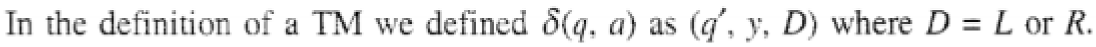
(iv) Whether to halt or not.

Definition: A Turing machine M is a 7-tuple, namely (Q, Σ, Γ, δ, q0, B, F), where

- Q is a finite nonempty set of states.
- Γ is a finite nonempty set of tape symbols,
- B is the blank.
- Σ is a nonempty set of input symbols and is a subset of Γ rand b ∉ Σ.
- δ is the transition function mapping (q, x) onto (q', y, D) where D denotes the direction of movement of R/W head D = L or R according as the movement is to the left or right.
- q0 ∈ Q is the initial state, and
- F ⊆ Q is the set of final states.

Techniques for TM Construction

1. **Turing Machine with Stationary Head :**



In the definition of a TM we defined $\delta(q, a)$ as (q', y, D) where $D = L$ or R.

The head moves to the left or right after reading an input symbol. Suppose , we want to include the option that the head can continue to be in same cell for some input symbol , then we define ό(q , a) as (q' , y, S)

This means that the TM , on reading the input symbol a , changes the state q' and writes y in the current cell in place of a and continues to remain in the same cell.

2. **Multiple Track Turing Machine**

In the case of TM defined earlier, a single tape was used.

In a multiple track TM. a single tape is assumed to be divided into several tracks. Now the tape alphabet is required to consist of k-tuples of tape symbols, k being the number of tracks.

Hence the only difference between the standard TM and the TM with multiple tracks is the set of tape symbols.

In the case of the standard Turing machine, **tape symbols are elements of Γ** in the case of TM with multiple track, **it is Γ^K**

3. **Storage in the State**

   We are using a state, whether it is of a FSM or PDA or TM, to 'remember' things.
   We can use a state to store a symbol as well. So, the state becomes a pair (q, a) where q is the state (in the usual sense) and 'a' is the tape symbol stored in (q, a). So, the new set of states becomes Q x Γ

4. **Subroutines**

First a TM program for the subroutine is written. This will have an initial state and a 'return' state. After reaching the return state. There is a temporary halt, for using a subroutine, new states are introduced.

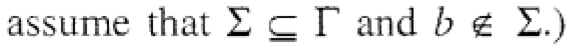
When there is a need for calling the subroutine, moves are affected to enter the initial state for the subroutine and to return to the main program of TM.

We use this concept to design a TM for performing multiplication of two positive integers.

Multitape Turing Machine :

A multitape TM has :

- a finite set Q of states.
- an initial state qo.
- a subset F of Q called the set of final states.
- a set P of tape symbols.
- a new symbol b, not in P called the blank symbol.



assume that $\Sigma \subseteq \Gamma$ and $b \notin \Sigma$.)

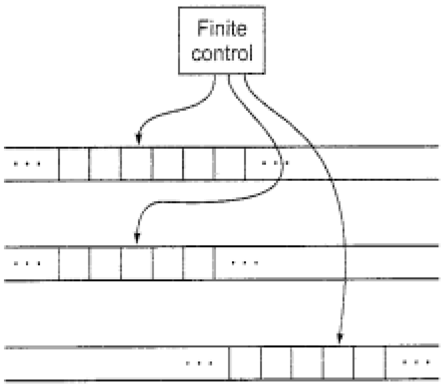


Fig. 9.8 Multitape Turing machine.

There are k tapes, each divided into cells.
The first tape holds the input string w.
Initially, all the other tapes hold the blank symbol.

Initially the head of the first tape (input tape) is at the left end of the input w.
All the other heads can be placed at any cell initially.

δ is a partial function from $Q \times \Gamma^K$ into $Q \times \Gamma^K \times \{L, R, S\}^K$.
A move depends on the current state and k tape symbols under k tape heads. I

In a typical move:

(i) M enters a new state.

(ii) On each tape, a new symbol is written in the cell under the head.

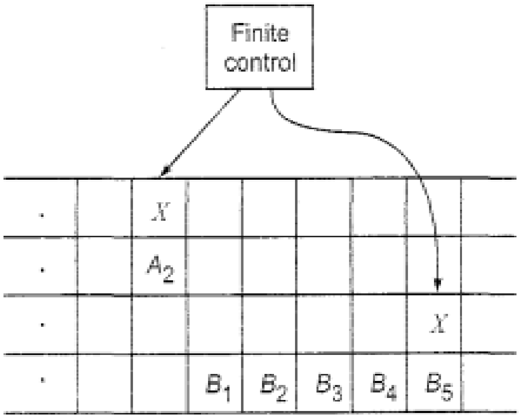
(iii) Each tape head moves to the left or right or remains stationary.
The heads move independently: some move to the left, some to the right and the remaining heads do not move.

- The initial ID has the initial state qo, the input string w in the first tape (input tape), empty strings of Blanks in the remaining k - 1 tapes.
- An accepting ID has a final state & some strings in each of the k tapes.

Theorem 9.1: Every language accepted by a multitape TM is acceptable by some single-tape TM (that is, the standard TM).

Proof: Suppose a language L is accepted by a k-tape TM M. We simulate M with a single-tape TM with 2k tracks.

The second. fourth, ..., (2k)th tracks hold the contents of the k-tapes. The first. third , ... , (2k - l)th tracks hold a head marker (a symbol say X) to indicate the position of the respective tape head.



Simulation of multitape TM.

Figure can be used to visualize the simulation.
The symbols A2 and B5 are the current symbols to be scanned and so the head marker X is above the two symbols.

Initially the contents of tapes 1 and 2 of M are stored in the second and fourth tracks of M1. The head markers of the first and third tracks are at the cells containing the first symbol.

To simulate a move of M, the 2k-track TM M1 has to visit the two head markers and store the scanned symbols in its control.

Keeping track of the head markers visited and those to be visited is achieved by keeping a count and storing it in the finite control of M1. The finite control of M1 has also the information about the states of M and its moves. After visiting both head markers. M1 knows the tape symbols being scanned by the two heads of M.

M1 revisits each of the head markers:

(i) It changes the tape symbol in the corresponding track of M1 based on the information regarding the move of M corresponding to the state (of M) and the tape symbol in the corresponding tape M.

(ii) It moves the head markers to the left or right.

(iii) M1 changes the state of M in its control. This is the simulation of a single move of M. At the end of this , M1 is ready to implement its next move based on the revised position of its head markers & the changed state available in its control.

This is the simulation of a single move of M.

Running time of M: The running time of M on input W, is the number of steps M takes before halting. If M does not halt on an input string W, then the running time of M on W is infinite

Time Complexity of M : Time complexity of M is the function T(n) , n being the input size
T(n) is defined as the maximum of the running time of M over all inputs W of size n.

Theorem 9.2: If M1 is the single-tape TM simulating multitape TM M, then the time taken by M1 to simulate n moves of M is O(n2).

Proof: Let M be a K-tape TM. After n moves, the head markers of M1 be separated by 2n cells or less.

To simulate a move of M, the TM M1 must visit all the K head markers.

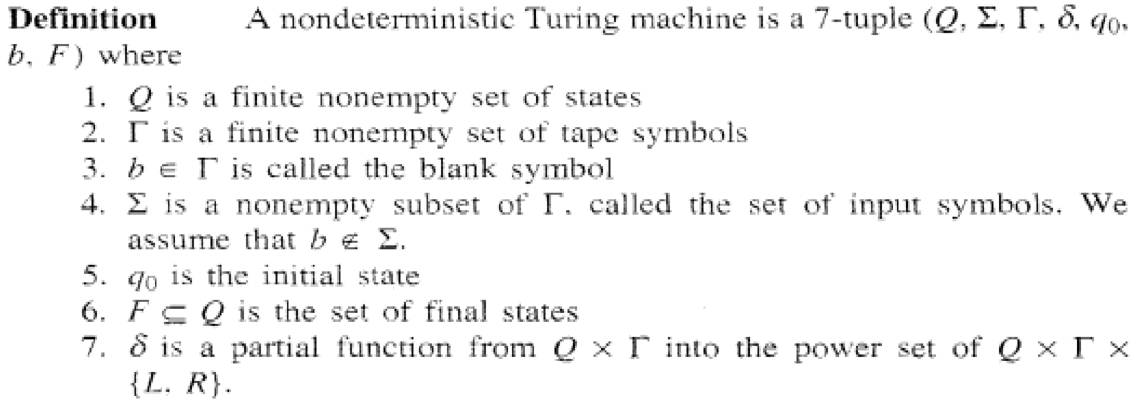
If M starts with the leftmost head marker, M1 will go through all the head markers by moving right by at most 2n cells.

To Simulate the changes in each tape , M1 has to move left by at most 2n cells; To simulate changes in K tapes , it requires at most two moves in the reverse direction for each tape.

The total number of moves by M , for simulating one move of M is at most **4n+2K .**

So the number of moves of M1 for simulating n moves of M1 for simulating n moves of M is **n(4n+2K)**. As the constant K is independent of n, the time taken by M1 is **O(n^2)**

Non Deterministic TM:



Definition A nondeterministic Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$ where

1. Q is a finite nonempty set of states
2. Γ is a finite nonempty set of tape symbols
3. $b \in \Gamma$ is called the blank symbol
4. Σ is a nonempty subset of Γ, called the set of input symbols. We assume that $b \notin \Sigma$.
5. q_0 is the initial state
6. $F \subseteq Q$ is the set of final states
7. δ is a partial function from $Q \times \Gamma$ into the power set of $Q \times \Gamma \times \{L, R\}$.

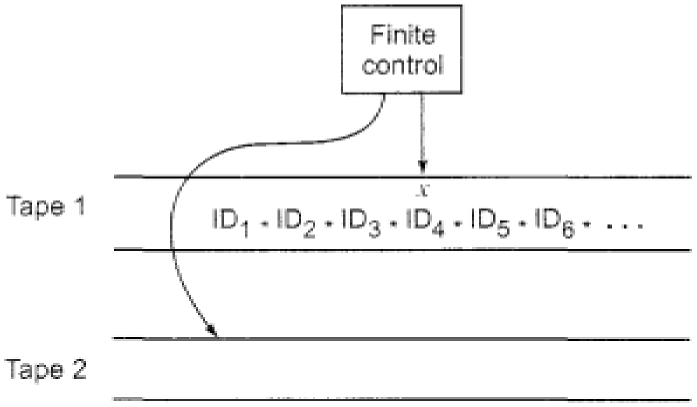
Theorem 9.3: If M is a nondeterministic TM, there is a deterministic TM M1 such that T(M) = T(M1)

Proof: We construct M1 as a multitape TM. Each symbol in the input string leads to a change in ID. M1 should be able to reach all IDs and stop when an ID containing a final state is reached.

So the first tape is used to store IDs of M as a sequence and also the state of M. These IDs are separated by the symbol *(included as a tape symbol).

The current ID is known by marking an x along with the ID-separator * (The symbol * marked with x is a new tape symbol.) All IDs to the left of the current one have been explored already and so can be ignored subsequently.

Note that the current ID is decided by the current input symbol of w.



The deterministic TM simulating *M*.

1. M1 examines the state and the scanned symbol of the current ID. Using the knowledge of moves of M stored in the finite control of M1, it checks whether the state in the current ID is an accepting state of M. In this case M1 accepts and stops simulating M.

2. If the state q say in the current ID xqay, is not an accepting state of M1 and δ(q, a) has k triples, M1 copies the ID xqay in the second tape and makes k copies of this ID at the end of the sequence of IDs in tape 2.
3. M1 modifies these k IDs in tape 2 according to the k choices given by δ(q, a).
4. M1 returns to the marked current ID. Erases the mark x and marks the next ID-separator * with x. Then M1 goes back to step 1.

LINEAR BOUNDED AUTOMATON

Linear bounded automaton (LBA) is a linear function used to restrict (to bound) the length of the tape. A linear bounded automaton is a nondeterministic Turing machine which has a single tape whose length is not infinite but bounded by a linear function of the length of the input string.

The models can be described formally by the following set format:

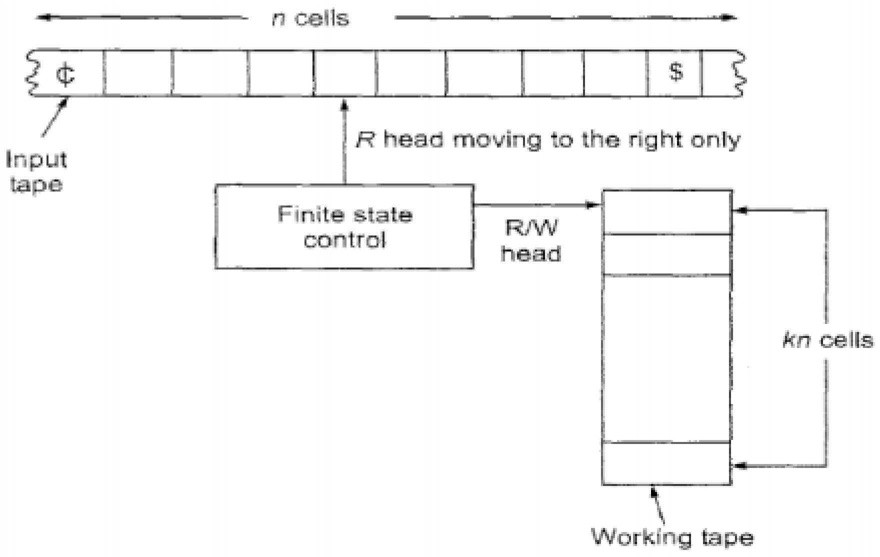
$$M = (Q, \Sigma, \Gamma, \delta, q_0, b, ¢, \$, F)$$

All the symbols have the same meaning as in the basic model of Turing machines with the difference that the input alphabet contains two special symbols **C and $.**
¢ is called the left-end marker which is entered in the leftmost cell of the input tape and prevents the R/W head from getting off the left end of the tape.
$ is called the right-end marker which is entered in the rightmost cell of the input tape and prevents the R/W head from getting off the right end of the tape.

Both the end markers should not appear on any other cell within the input tape, and the RIW head should not print any other symbol over both the end markers.

Let us consider the input string w with |w|= n-2.
The input string w can be recognized by an LBA if it can also be recognized by a Turing machine using no more than kn cells of input tape, where k is a constant specified in the description of LBA.

The value of k does not depend on the input string but is purely a property of the machine. Whenever we process any string in LBA, we shall assume that the input string is enclosed within the end markers ¢ and $.



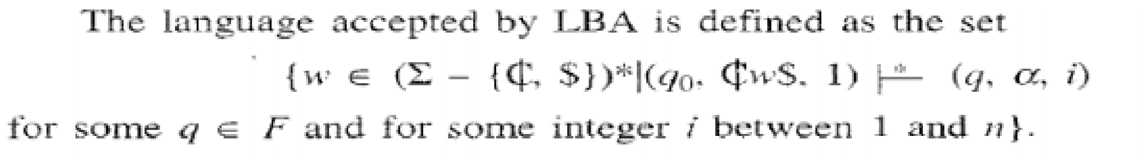
Model of linear bounded automaton.

There are two tapes

One is called the **input tape**, and the other, **working tape**.

- On the input tape the head never prints and never moves to the left.
- On the working tape the head can modify the contents in any way, without any restriction.

In the case of LBA, an ID is denoted by (q, w. k), where q € Q. w € Γ and k is some integer between 1 and n. The transition of IDs is similar except that k changes to k - 1 if the R/W head moves to the left and to k + 1 if the head moves to the right.



The language accepted by LBA is defined as the set

$$\{w \in (\Sigma - \{¢, \$\})^* | (q_0, ¢w\$, 1) \vdash^{*} (q, \alpha, i)$$

for some $q \in F$ and for some integer i between 1 and $n\}$.

Relation between LBA and context-sensitive languages:

The set of strings accepted by nondeterministic LBA is the set of strings generated by the context-sensitive grammars, excluding the null strings.

Now we give an important result:

If L is a context-sensitive language, then L is accepted by a linear bounded automaton. The converse is also true.

Definition of an algorithm :

An algorithm is defined as step by step procedure to solve a given problem.
The procedure is finite sequence of instructions that have to be executed to complete a task.
The algorithm is terminated after finite number of steps for any input .

Decidability :

As an algorithm terminates eventually, the TM also terminates.
The TM halts in following two situations :

When a TM reaches a final state , it halts

When a TM reaches a state q , when the scanned input symbol is 'a' and if the transition δ (q , a) is not defined , it halts.

But , there are some TM that never halt on some inputs in any of the above situations. So we have to make a distinction between the languages that are accepted by TM & halts on all inputs & a TM that never halts on some input strings.

Recursively Enumerable Languages :

A language $L \subseteq \sum^*$ is recursively enumerable iff there exists a TM M such that L=T(M) where T(M) is the language accepted by TM.

Recursive Language :

A Language L $L \subseteq \sum^*$ is recursive iff there exists a TM that satisfies the following two conditions :
If w belongs to L , then W is accepted by TM on reaching the final state & the machine halts.
If W does not belong to L , then W is rejected by TM without reaching the final state & the machine halts .

Decidable languages : A problem with two answers (Yes / No) is decidable if the corresponding language is recursive. In this case , the language L is also called decidable .

Undecidable language : A problem /language is undecidable if it is not decidable .

Undecidability

Theorem 10.4 There exists a language over L: that is not recursively enumerable.

Proof A language L is recursively enumerable if there exists a TM M such that $L = T(M)$. As Σ is finite, Σ^* is countable (that is, there exists a one-to-one correspondence between Σ^* and N).

As a Turing machine M is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$ and each member of the 7-tuple is a finite set, M can be encoded as a string. So the set I of all TMs is countable.

Let $\mathscr{L}$ be the set of all languages over Σ. Then a member of $\mathscr{L}$ is a subset of Σ^* (Note that Σ^* is infinite even though Σ is finite). We show that $\mathscr{L}$ is uncountable (that is, an infinite set not in one-to correspondence with N).

We prove this by contradiction. If $\mathscr{L}$ were countable then $\mathscr{L}$ can be written as a sequence $\{L_1, L_2, L_3, \ldots\}$. We write Σ^* as a sequence $\{w_1, w_2, w_3, \ldots\}$. So L_i can be represented as an infinite binary sequence $x_{i1}x_{i2}x_{i3}\ldots$ where

$$x_{ij} = \begin{cases} 1 & \text{if } w_j \in L_i \\ 0 & \text{otherwise} \end{cases}$$

Using this representation we write L_i as an infinite binary sequence.

$$\begin{array}{llll}
L_1 : & x_{11}x_{12}x_{13} & \ldots\ x_{1j} & \ldots \\
L_2 : & x_{21}x_{22}x_{23} & \ldots\ x_{2j} & \ldots \\
\vdots & & \quad\ \vdots & \\
L_i : & x_{i1}x_{i2}x_{i3} & \ldots\ x_{ij} & \ldots
\end{array}$$

Fig. 10.1 Representation of $\mathcal{L}$.

We define a subset L of Σ^* by the binary sequence $y_1y_2y_3 \ldots$ where $y_i = 1 - x_{ii}$. If $x_{ii} = 0$, $y_i = 1$ and if $x_{ii} = 1$, $y_i = 0$. Thus according to our assumption the subset L of Σ^* represented by the infinite binary sequence $y_1y_2y_3 \ldots$ should be L_k for some natural number k. But $L \neq L_k$, since $w_k \in L$ if and only if $w_k \notin L_k$. This contradicts our assumption that $\mathcal{L}$ is countable. Therefore $\mathcal{L}$ is uncountable. As I is countable. $\mathcal{L}$ should have some members not corresponding to any TM in I. This proves the existence of a language over Σ tnat is not recursively enumerable. ∎

Definition 10.8 $A_{\text{TM}} = \{(M,\ w) \mid \text{The TM } M \text{ accepts } w\}$.

Theorem 10.5 A_{TM} is undecidable.

Proof We can prove that A_{TM} is recursively enumerable. Construct a TM U as follows:

$(M,\ w)$ is an input to U. Simulate M on w. If M enters an accepting state, U accepts $(M,\ w)$. Hence A_{TM} is recursively enumerable. We prove that A_{TM} is undecidable by contradiction. We assume that A_{TM} is decidable by a TM H that eventually halts on all inputs. Then

$$H(M,\ w) = \begin{cases} \text{accept} & \text{if } M \text{ accepts } w \\ \text{reject} & \text{if } M \text{ does not accept } w \end{cases}$$

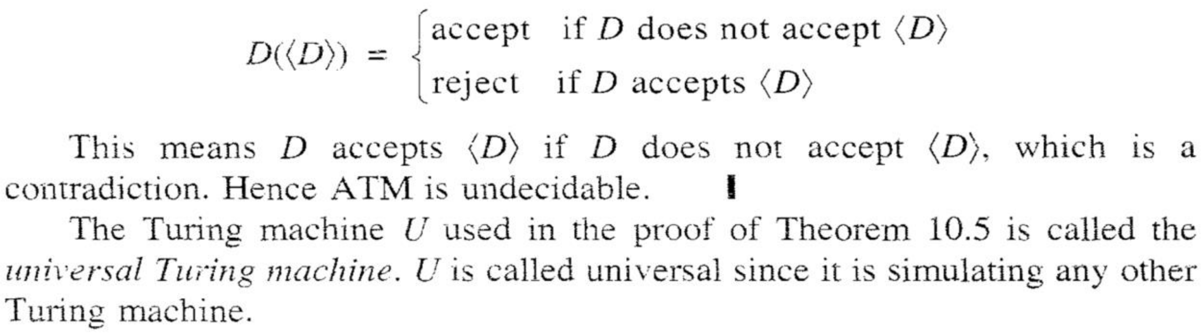
We construct a new TM D with H as subroutine. D calls H to determine what M does when it receives the input $\langle M \rangle$, the encoded description of M as a string. Based on the received information on $(M,\ \langle M \rangle)$, D rejects M if M accepts $\langle M \rangle$ and accepts M if M rejects $\langle M \rangle$. D is described as follows:

1. $\langle M \rangle$ is an input to D, where $\langle M \rangle$ is the encoded string representing M.
2. D calls H to run on $(M,\ \langle M \rangle)$
3. D rejects $\langle M \rangle$ if H accepts $(M,\ \langle M \rangle)$ and accepts $\langle M \rangle$ if H rejects $(M,\ \langle M \rangle)$.

Now step 3 can be described as follows:

$$D(\langle M \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ does not accept } \langle M \rangle \\ \text{reject} & \text{if } M \text{ accepts } \langle M \rangle \end{cases}$$

Let us look at the action of D on the input $\langle D \rangle$. According to the construction of D,



$$D(\langle D\rangle) = \begin{cases}\text{accept} & \text{if } D \text{ does not accept } \langle D\rangle\\ \text{reject} & \text{if } D \text{ accepts } \langle D\rangle\end{cases}$$

This means D accepts $\langle D\rangle$ if D does not accept $\langle D\rangle$, which is a contradiction. Hence ATM is undecidable. ▮

The Turing machine U used in the proof of Theorem 10.5 is called the *universal Turing machine*. U is called universal since it is simulating any other Turing machine.

HALTING PROBLEM :

Given a TM M & an input string W with the initial configuration q0 , after some (or all) computations do the machine M halts ?

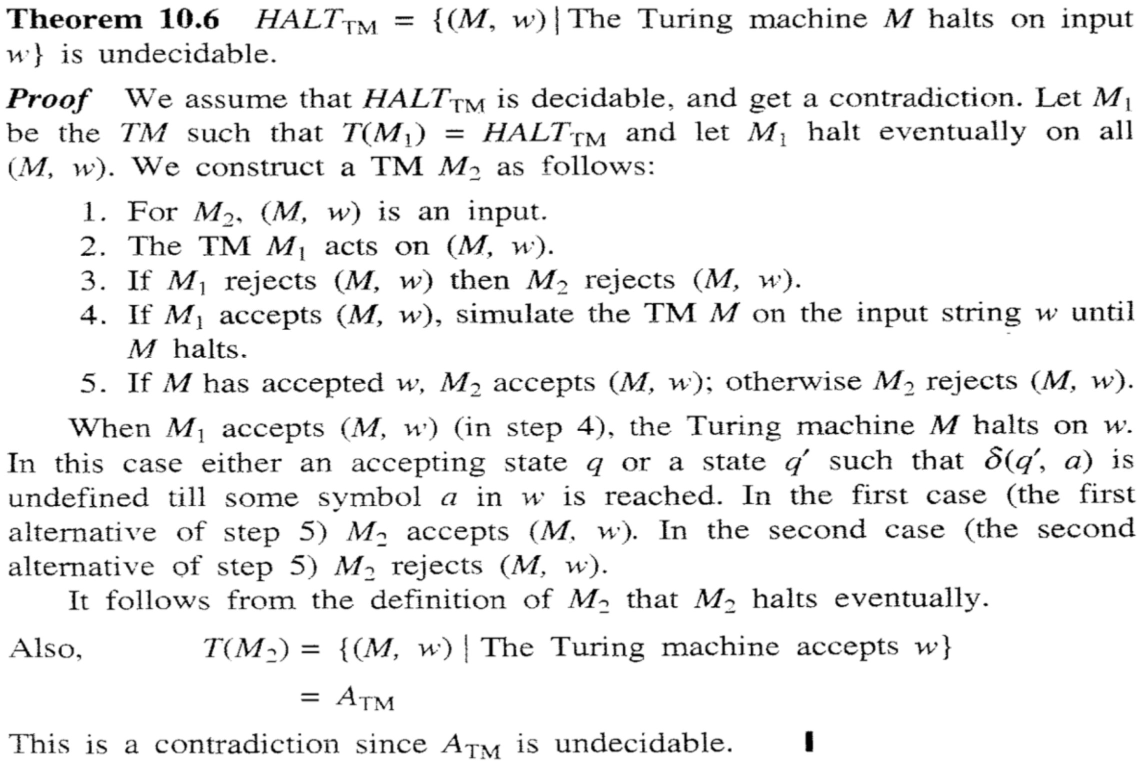
In other words , if M is a TM , we have to identify whether (M,W) halts or does not halt when W is applied as the input .

Given the description of an arbitrary TM M & the input string W , we are looking for a single TM that will predict whether or not the computation of M applied applied to W will halt.

A reduction technique is used to prove the undecidability of turing problem of a TM. Using this technique , a problem A is reducible to problem B if a solution to the problem B can be used to solve the problem A. Thus ,

If A is reducible to B & B is decidable , then A is decidable .

If A is reducible to B & B is undecidable , then A is Undecidable.



Theorem 10.6 $HALT_{TM} = \{(M, w) \mid \text{The Turing machine } M \text{ halts on input } w\}$ is undecidable.

Proof We assume that $HALT_{TM}$ is decidable, and get a contradiction. Let M_1 be the TM such that $T(M_1) = HALT_{TM}$ and let M_1 halt eventually on all (M, w). We construct a TM M_2 as follows:

1. For M_2, (M, w) is an input.
2. The TM M_1 acts on (M, w).
3. If M_1 rejects (M, w) then M_2 rejects (M, w).
4. If M_1 accepts (M, w), simulate the TM M on the input string w until M halts.
5. If M has accepted w, M_2 accepts (M, w); otherwise M_2 rejects (M, w).

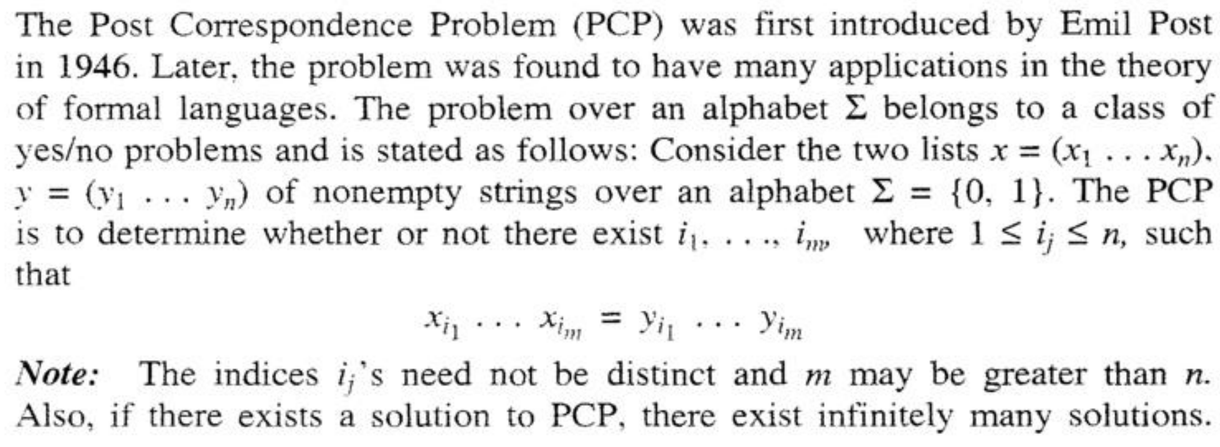
When M_1 accepts (M, w) (in step 4), the Turing machine M halts on w. In this case either an accepting state q or a state q' such that $\delta(q', a)$ is undefined till some symbol a in w is reached. In the first case (the first alternative of step 5) M_2 accepts (M, w). In the second case (the second alternative of step 5) M_2 rejects (M, w).

It follows from the definition of M_2 that M_2 halts eventually.

Also, $$T(M_2) = \{(M, w) \mid \text{The Turing machine accepts } w\}$$
$$= A_{TM}$$

This is a contradiction since A_{TM} is undecidable. ▮

THE POST CORRESPONDENCE PROBLEM



The Post Correspondence Problem (PCP) was first introduced by Emil Post in 1946. Later, the problem was found to have many applications in the theory of formal languages. The problem over an alphabet Σ belongs to a class of yes/no problems and is stated as follows: Consider the two lists $x = (x_1 \ldots x_n)$, $y = (y_1 \ldots y_n)$ of nonempty strings over an alphabet $\Sigma = \{0, 1\}$. The PCP is to determine whether or not there exist $i_1, \ldots, i_m$, where $1 \leq i_j \leq n$, such that

$$x_{i_1} \ldots x_{i_m} = y_{i_1} \ldots y_{i_m}$$

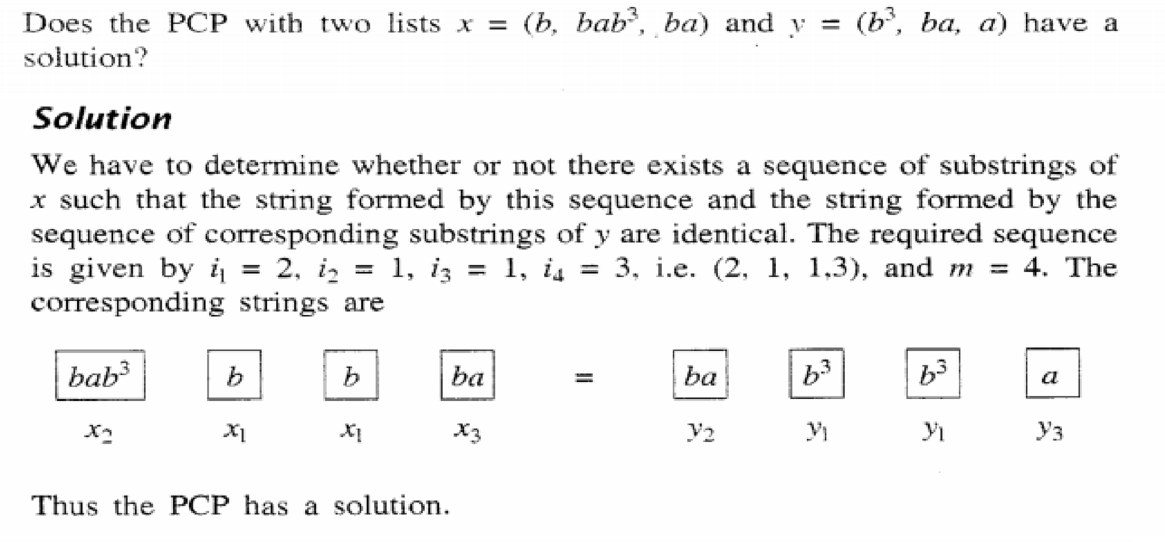
Note: The indices i_j's need not be distinct and m may be greater than n. Also, if there exists a solution to PCP, there exist infinitely many solutions.

A post correspondence system consists of a finite set of ordered pairs (x_i, y_i), $i = 1, 2, \cdots, n$, where $x_i, y_i \in \Sigma^+$ for some alphabet Σ.

Any sequence of numbers $i_1, i_2, \cdots, i_k$ $s-t.$

is called a solution to a Post Correspondence System.

$x_{i_1}, x_{i_2}, \cdots, x_{i_k} = y_{i_1}, y_{i_2}, \cdots, y_{i_k}$ The Post's Correspondence Problem is the problem of determining whether a Post Correspondence system has a solutions.



Does the PCP with two lists $x = (b, bab^3, ba)$ and $y = (b^3, ba, a)$ have a solution?

Solution

We have to determine whether or not there exists a sequence of substrings of x such that the string formed by this sequence and the string formed by the sequence of corresponding substrings of y are identical. The required sequence is given by $i_1 = 2$, $i_2 = 1$, $i_3 = 1$, $i_4 = 3$, i.e. (2, 1, 1,3), and $m = 4$. The corresponding strings are

$$\underset{x_2}{\boxed{bab^3}}\quad \underset{x_1}{\boxed{b}}\quad \underset{x_1}{\boxed{b}}\quad \underset{x_3}{\boxed{ba}}\quad = \quad \underset{y_2}{\boxed{ba}}\quad \underset{y_1}{\boxed{b^3}}\quad \underset{y_1}{\boxed{b^3}}\quad \underset{y_3}{\boxed{a}}$$

Thus the PCP has a solution.

Problems

Construct Turing machine for the following languages.

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

deterministic machine only for valid string can't ~~no need~~ to show for Invalid

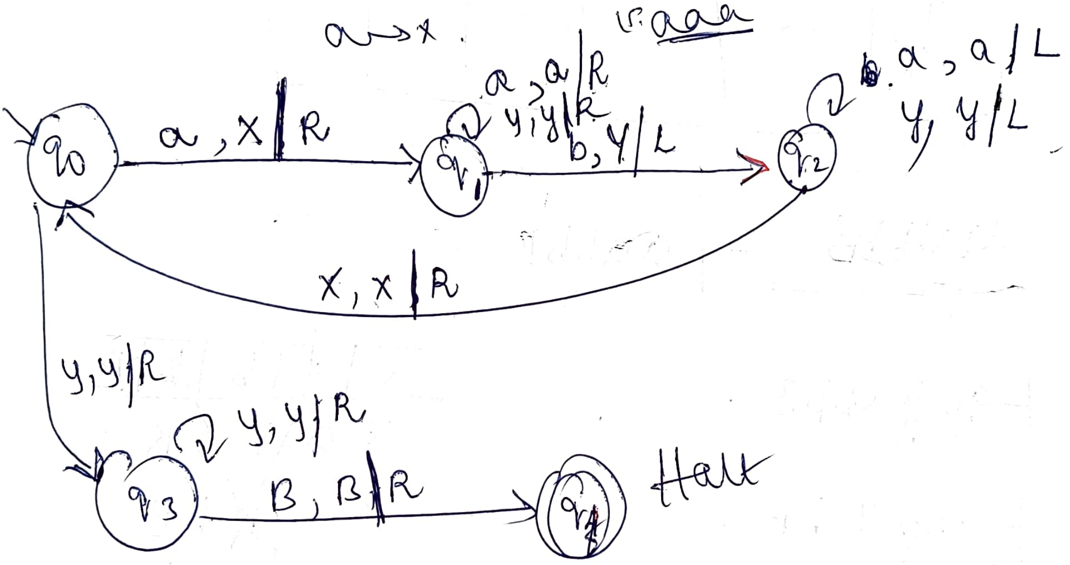
Soln: a a a bbb B..

B	B	a	a	a	b	b	b	B

#a = #b. Take aabb.

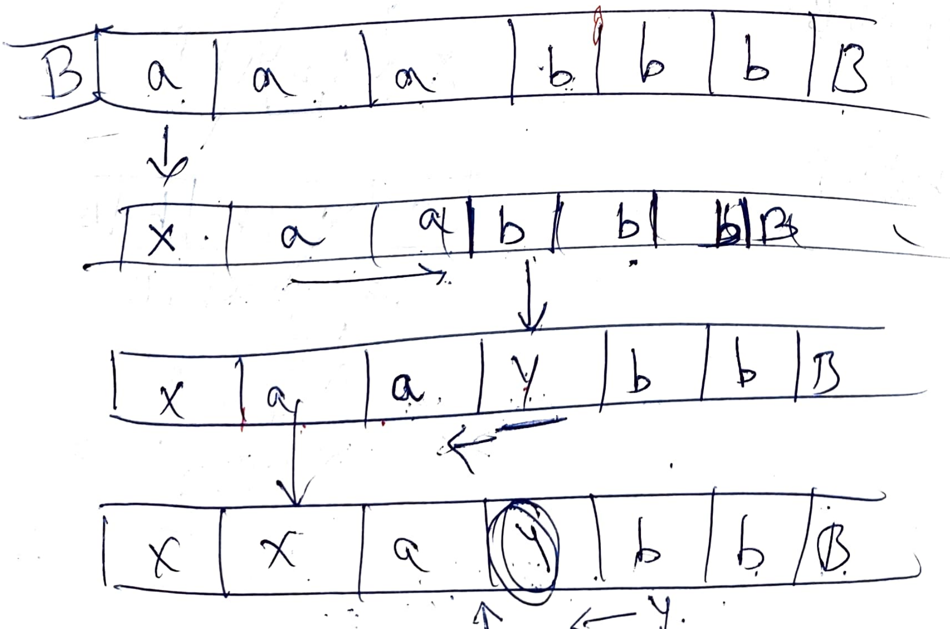
a = x , b = y. a Corresponding b

a → x. i: aaa



n = 1 ab

X → Y



ID:

a	a	b	b	B

$\vdash \underbrace{q_0 \boxed{a}}\underline{a}bbB$

Value state

$q\,x\underline{a}bbB$

$\boxed{x\,q\,abbB}$

$\vdash X\underbrace{q_1 a}bbB$

$\vdash X a \underbrace{q_1 b} bB$

$\delta(q_1, b)$

~~$\vdash X a Y q_2 bB$~~

q_2

$\vdash X \underbrace{q_2 a} Y bB$

First write the state & then shange the direction.

$\vdash \underbrace{q_2 X} a Y bB$

$\vdash X \underbrace{q_0 a} Y bB$

Make a note of.

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

$\vdash X X \underbrace{q_1 Y} bB$

$\vdash X X Y \underbrace{q_1 b} B$

$\vdash X X \underbrace{q_2 Y} Y B$

$\vdash X \underbrace{q_2 X} Y Y B$

$\vdash X X \underbrace{q_0 Y} Y B$

~~$\vdash X X Y \underbrace{q_0 Y} B$~~

~~$\vdash X X Y Y q_0 B$~~

~~$\vdash X X Y Y B q_3$~~ EF

$\vdash X X Y \underbrace{q_3 Y} B$

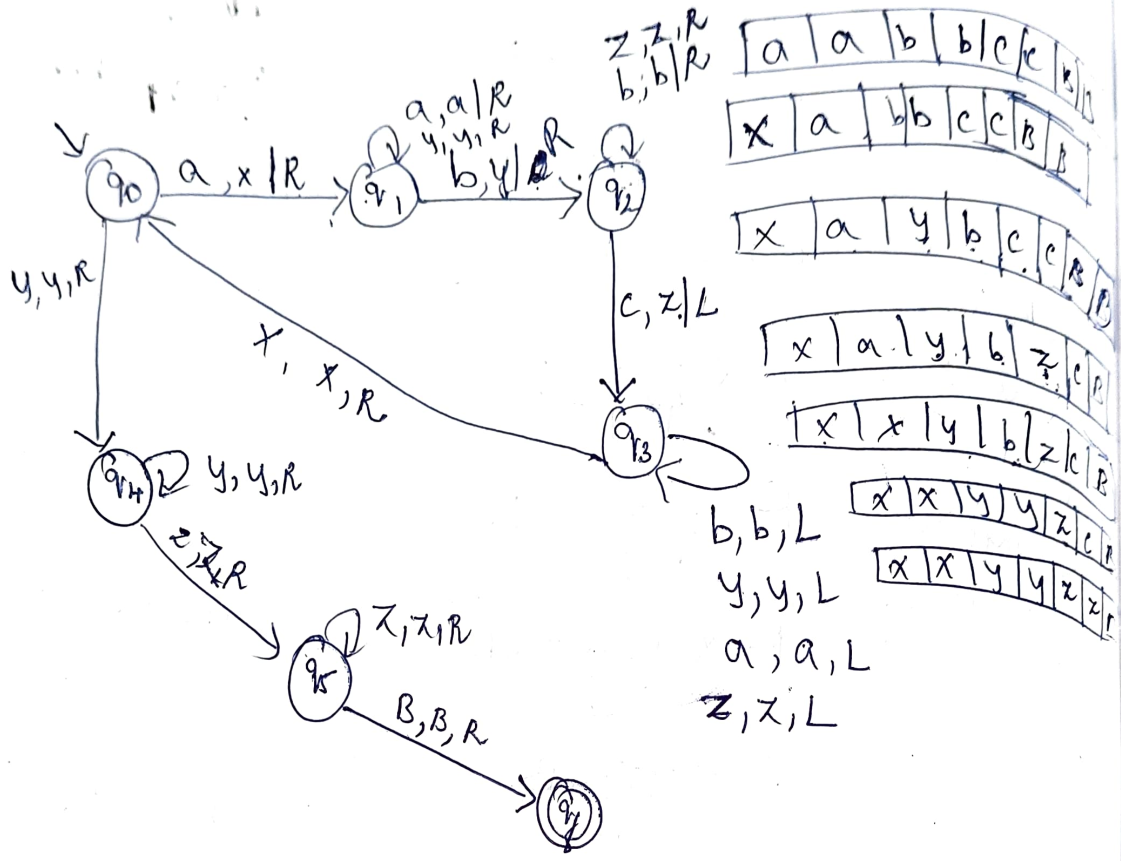
$\vdash X X Y Y \underbrace{q_3 B}$

$\vdash X X Y Y q_f$

EF

2) $L = \{a^n b^n c^n \mid n \geq 1\}$ | $L = \{a^n b^n c^n \mid n > 0\}$

a	a	b	b	c	c	B	B



a, x | R
a, a | R
y, y, R
b, y | R
Z, Z, R
b, b | R
c, Z | L
x, x, R
y, y, R
y, y, R
Z, Z, R
Z, Z, R
B, B, R
b, b, L
y, y, L
a, a, L
Z, Z, L
q0
q1
q2
q3
q4
q5
q6
| a | a | b | b | c | c | B | B
| x | a | b | b | c | c | B | B
| x | a | y | b | c | c | B | B
| x | a | y | b | z | c | B
| x | x | y | b | z | c | B
| x | x | y | y | z | c | B
| x | x | y | y | z | z | B

Transition Table.

	a	b	x	y	z	B
q0	(q1, x R)	—	—	q4, y, R	—	—
q1	q1, a R	q2, y R	—	q1, y R	—	—
q2	—	q2 b R	—	—	q2 Z R	—
q3	q3, a L	q3 b L	q0 x R	q3 y L	q3 Z L	—
q4	—	—	—	q4 y R	q5 Z R	—
q5	—	—	—	—	q5 Z R	q6 B, R
q6	—	—	—	—	—	—

Ip: aabbcc

$\vdash \underbrace{q_0 a}abbccB$

$\vdash X \underbrace{q_1 a}bbccB.$

$\vdash X a \underbrace{q_1 b}bccB$

$\vdash X a Y \underbrace{q_2 b}ccB$

$\vdash X a Y b \underbrace{q_2 c}cB$

$\vdash X a \underbrace{\overset{}{y} q_3 b} Z\, cB$

$\vdash X a \underbrace{q_3 Y} b Z cB$

$\vdash X \underbrace{q_3 a} Y b Z cB$

$\vdash \underbrace{q_3 X} a Y b Z cB$ ✓

$\vdash X \underbrace{q_0 a} Y.b Z cB.$

$\vdash X X \underbrace{q_1 Y} b Z cB$ ✓

$\vdash X X Y \underbrace{q_1 b} Z CB$ ✓

$\vdash X X Y Y \underbrace{q_2 Z} CB$ ✓

$\vdash X X Y Y Z \underbrace{q_2 c} B$ ✓

$\vdash X X Y Y Z Z \underbrace{q_3 B}$

$\vdash X X \underbrace{q_0 Y} Y Z Z B$

$\vdash X X Y q_4 Y Z Z B$

$X X Y Y q_3 Z Z B$

$X X Y q_3 Y Z Z B$

$X X q_3 Y Y Z Z B$

$X q_3 X Y Y Z Z B$

$X X q_0 Y Y Z Z B$

$X X Y q_4 Y Z Z B$

$X X Y Y q_4 Z Z B$

$X X Y Y Z q_5 Z B$

$X X Y Y Z Z q_5 B$

$X X Y Y Z Z B q_f$

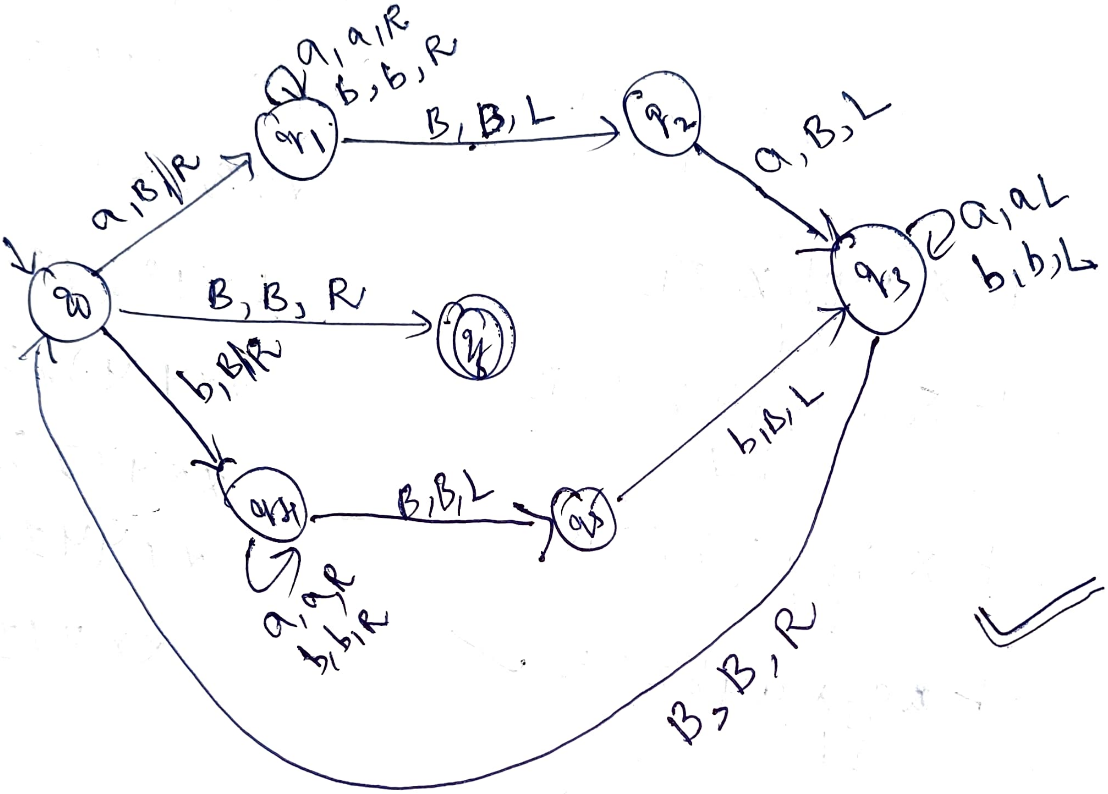
$\vdash X X Y Y Z Z q_4 B$

$\vdash X X Y Y \cdot Z Z q_5 CF$

3) $L = \{ww^R \mid w \in \{a,b\}^*\}$ Even Palindrome ~~[illegible]~~.

↓ Same empty also

$\underbrace{a\,b\,a\,b}_{w}\;\underbrace{b\,a\,b\,a}_{w^R}\,B$ → even length



B a b a b b a b a B

a → B

B	B	a	b	a	b	b	a	b	a	B	B

↑B

	B	b	a	b	b	a	b	a	B

	B	b	a	b	b	a	b	B	B

B ab ab baba B
↑

Jyothsna M S

abbbba

BBb ab babaB.
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑.
←
↑.

BBbabbabBB.
↑
↓
BB B abb

$\vdash q_0$ ababbabaB.

$\vdash$ B q_1 B babbabaB

3) $L = \{ww^R \mid w \in \{a,b\}^*\}$ Even palindum

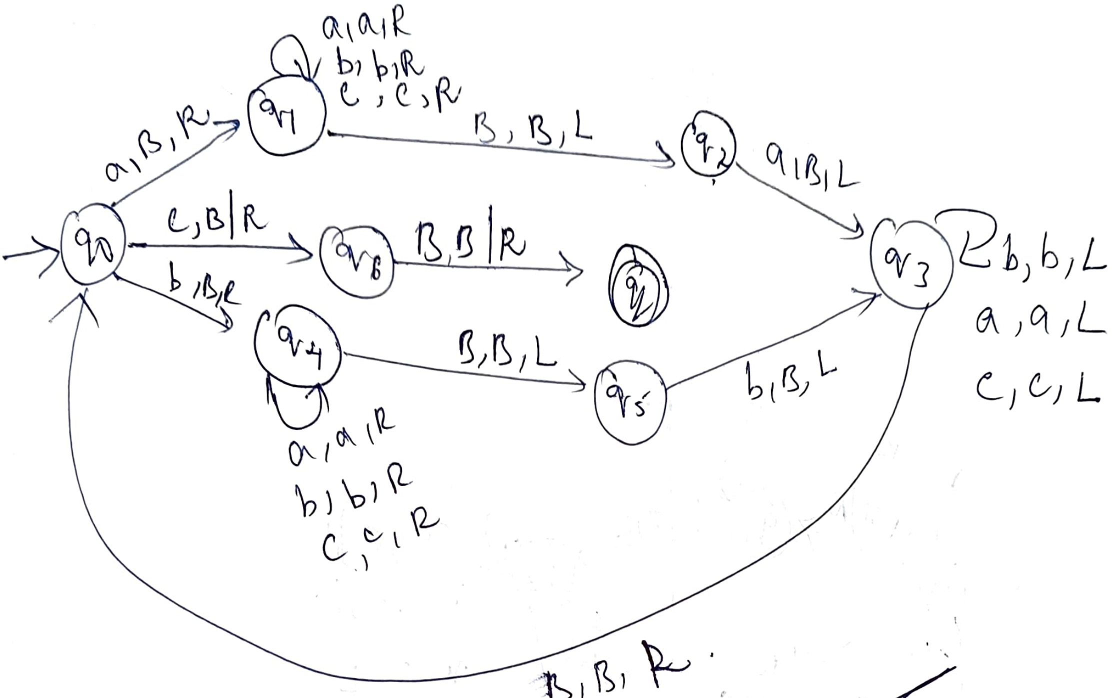
ababbabaB
x babbab~~a~~
x babbabx
x y abba y x
x y x bba y x
x y x y b ~~x~~ y x.
x y x y y x y x

4) Odd palindrome

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

A *

ab ab c b a b a B



⊢ q0 ababcbabaB.

$n_0 = n_1$

$n_a = n_b$.

Number of a's = number of b's.

(a) no. of 0's = no. of 1's.

B110010B

ab

B11X010B

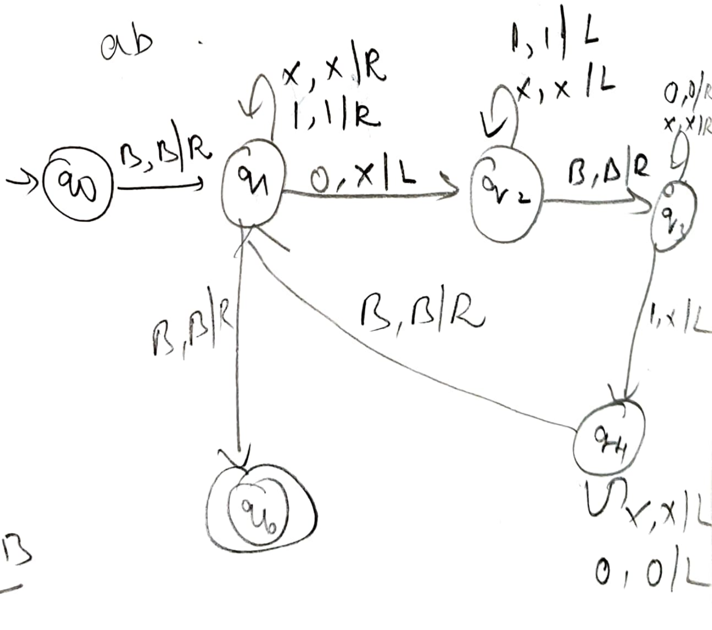
BX1X010B

BX1XX10B

BXXXX10B

BXXXXX0B

BXXXXXXB



1,1|L
x,x|L
x,x|R
1,1|R
0,0|R
x,x|R
q0
B,B|R
q1
0,x|L
q2
B,B|R
q3
B,B|R
B,B|R
1,x|L
q4
x,x|L
0,0|L
q6