4.1 DECIDABLE AND UNDECIDABLE QUESTIONS OF CONTEXT FREE LANGUAGES

4.1.1 Decidable questions under CFL:

1. **Membership:**

   Given a language L and a string w, is w in L?

2. **Emptiness and Finiteness:**

   Given a context-free language *L*. there exists a decision procedure that answers each of the following questions:

   1. Given a context-free language L. is L = Φ?
   2. Given a context-free language L. is L infinite?

3. **Equivalence:**

   Given two deterministic context-free languages L1 und L2, there exists a decision procedure to determine whether L1 = L1.

4.1.2Undecidable questions under CFL:

1. Given a context free language L, is L = Σ*?
2. Given a context-free language L, is the complement of L context-free?
3. Given a context-free language L, is L regular?
4. Given two context-free languages L1 and L2, is L1 = L2?
5. Given two context-free languages L1 and L2, is L1⊆ L2?
6. Given two context-free languages L1 and L2, is L1 ∩ L2= Φ?
7. Given a context-free language L. is L inherently ambiguous?
8. Given a context-free grammar G. is G ambiguous?

4.2 TURING MACHINES

Turing formulated a model of algorithm or computation, that is widely accepted. The Church-Turing thesis states that **"Any algorithmic procedure that can be carried out by human beings/computer can be carried out by a Turing machine"**. It has been universally accepted by computer scientists that the Turing machine provides an ideal theoretical model of a computer.
For formalizing computability, Turing assumed that, while computing, a person writes symbols on a one-dimensional tape which is divided into cells. One scans the cells one at a time and usually performs one of the three simple operations, namely

(i) writing a new symbol in the cell being currently scanned,
(ii) moving to the cell left of the present cell and
(iii) moving to the cell right of the present cell.

With these observations in mind, Turing proposed his **'computing machine.'**

4.2.1 Turing machine Model

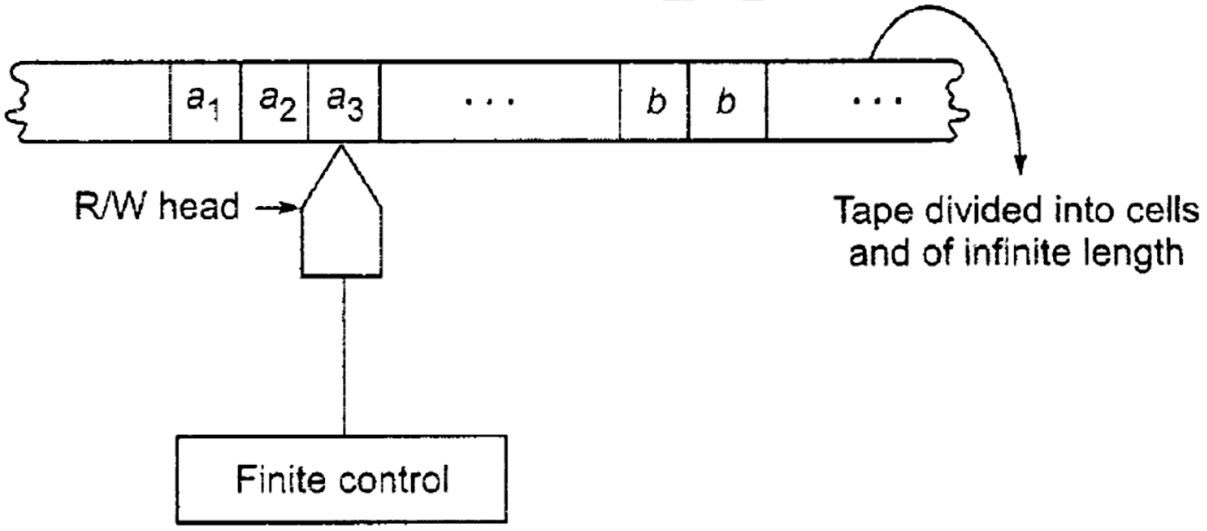


Fig. Turing Machine model

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.

4.2.2 Formal definition of Turing Machine

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.

4.3 REPRESENTATION OF TURING MACHINE

We can describe a Turing machine employing

(i) Transition diagram (transition graph).
(ii) Instantaneous descriptions using move-relations.
(iii) Transition table

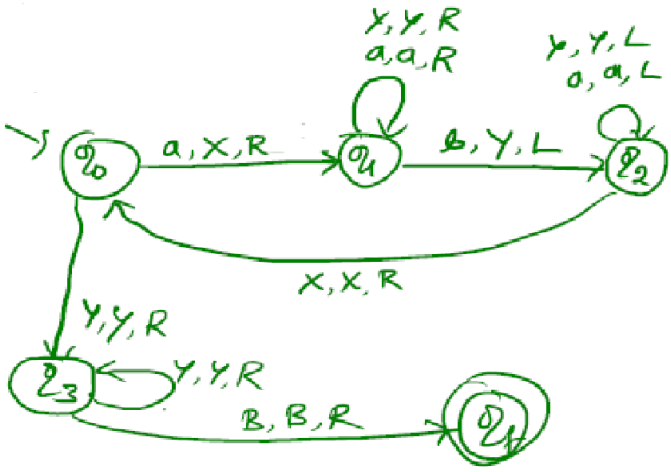
4.4 TURING MACHINE PROBLEMS

Problem 1: Design TM that accepts $\{a^n b^n \mid n \geq 1\}$

1. TRANSITIONS

$(q_0, a) \rightarrow (q_1, X, R)$
$(q_1, a) \rightarrow (q_1, a, R)$
$(q_1, b) \rightarrow (q_2, Y, L)$
$(q_2, a) \rightarrow (q_2, a, L)$
$(q_2, X) \rightarrow (q_0, X, R)$
$(q_1, Y) \rightarrow (q_1, Y, R)$
$(q_2, Y) \rightarrow (q_2, Y, L)$
$(q_0, Y) \rightarrow (q_3, Y, R)$
$(q_3, Y) \rightarrow (q_3, Y, R)$
$(q_3, B) \rightarrow (q_f, B, R)$

2. TRANSITION DIAGRAM



3. TRANSITION TABLE

	a	b	X	Y	B
→ q_0	q_1, X, R	H	H	q_3, Y, R	H
q_1	q_1, a, R	q_2, Y, L		q_1, Y, R	
q_2	q_2, a, L		q_0, X, R	q_2, Y, L	
q_3				q_3, Y, R	q_f, B, R
* q_f					

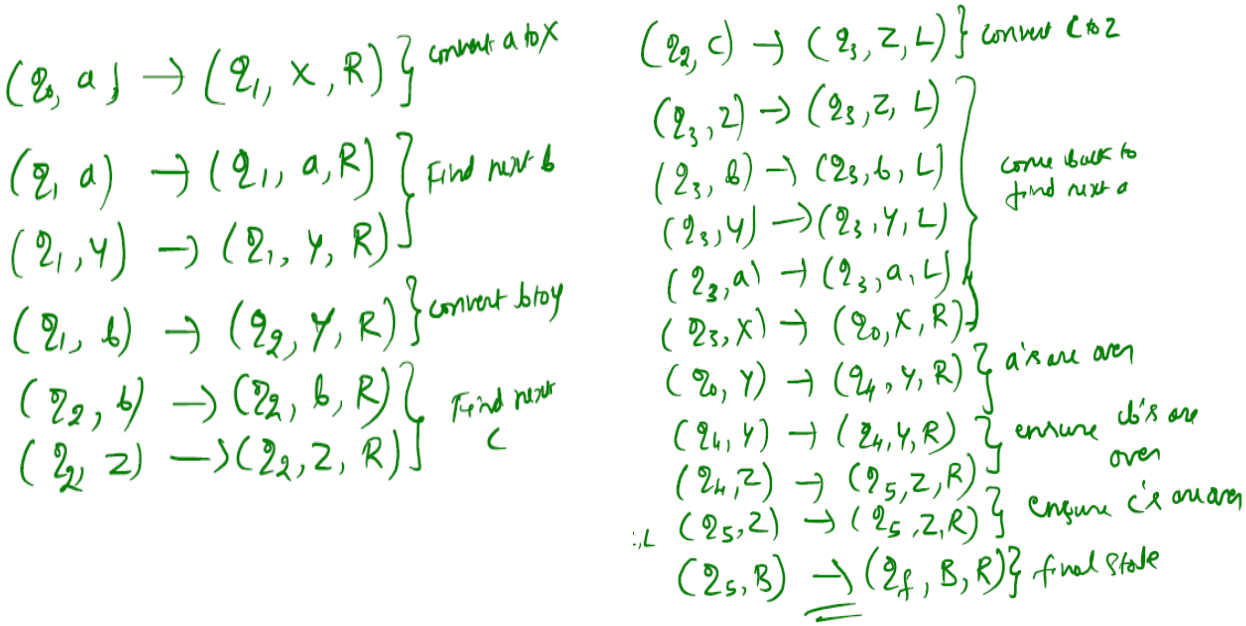
Turing machine is constructed for recognizing the language by converting each ‘a’ in to ‘X’ (state q0) and then finding the next occurrence of ‘b’ by moving to the right of the tape (state q1), passing through a’s and previously converted Y’s, once ‘b’ is found, it will be converted to Y and control moves towards left to find the next ‘a’(state q2), again passing through Y’s and a’s. This will continue in loop for all occurrence of a’s and matching b’s. If it doesn’t find an a, then it moves towards right to check there are no b’s left (state q3), if there are no b’s left, read write head reads a blank and hence halts by moving to final state (state qf).

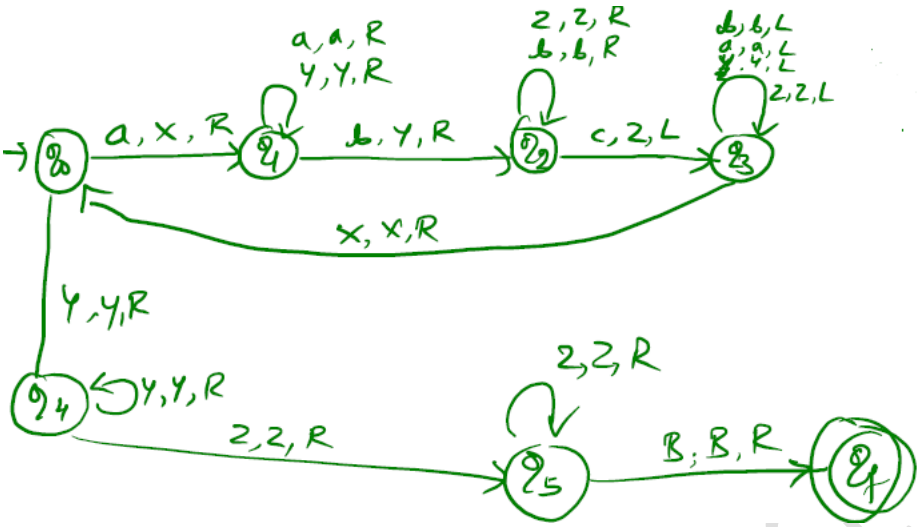
Write the Instantaneous Description (ID) for the string ‘aaabbb’

q_0 aaabbb |- X$q_1$aabbb|- Xa$q_1$abbb|- Xaa$q_1$bbb|- Xaq_2aYbb|- X$q_2$aaYbb|- q_2XaaYbb|- X$q_0$aaYbb |- XXq_1aYbb |- XXaq_1Ybb |- XXaY$q_1$bb |- XXaq_2YYb |- XXq_2aYYb |- Xq_2XaYYb |- XXq_0aYYb |- XXXq_1YYb |- XXXYq_1Yb |- XXXYYq_1b |- XXXYq_2YY |- XXXq_2YYY |- XXq_2XYYY |- XXXq_0 YYY |- XXXYq_3 YY |- XXXYYq_3 Y |- XXXYYYq_3B |- XXXYYYBq_f

Problem2:

Design TM that accepts {$a^nb^nc^n$| n≥ 1 }





Problem 3: Obtain a Turing machine to accept palindrome consisting of a's and b's.

$(q_0, a) \rightarrow (q_1, B, R)$
$(q_1, a) \rightarrow (q_1, a, R)$
$(q_1, b) \rightarrow (q_1, b, R)$
$(q_1, B) \rightarrow (q_2, B, L)$
$(q_2, a) \rightarrow (q_3, B, L)$
} matching 'a's

$(q_3, a) \rightarrow (q_3, a, L)$
$(q_3, b) \rightarrow (q_3, b, L)$
$(q_3, B) \rightarrow (q_0, B, R)$
} To find next Symbol

$(q_0, b) \rightarrow (q_4, B, R)$
$(q_4, a) \rightarrow (q_4, a, R)$
$(q_4, b) \rightarrow (q_4, b, R)$
$(q_4, B) \rightarrow (q_5, B, L)$
$(q_5, b) \rightarrow (q_3, B, L)$
} matching b's

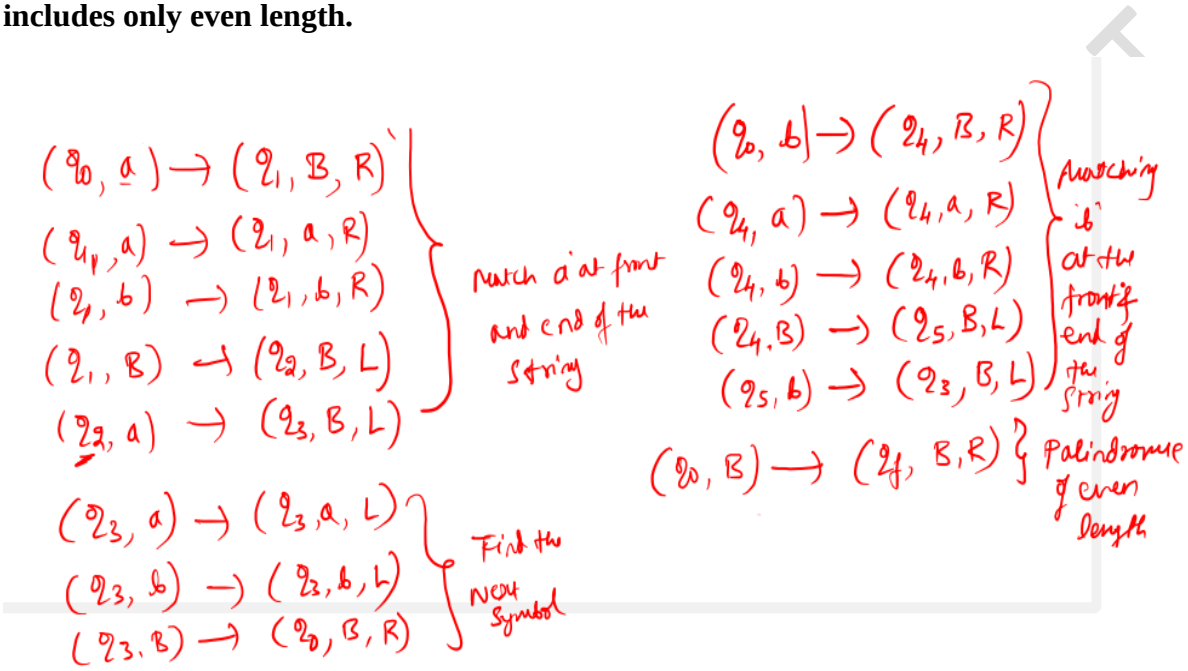
$(q_0, B) \rightarrow (q_f, B, R)$ } even length palindrome

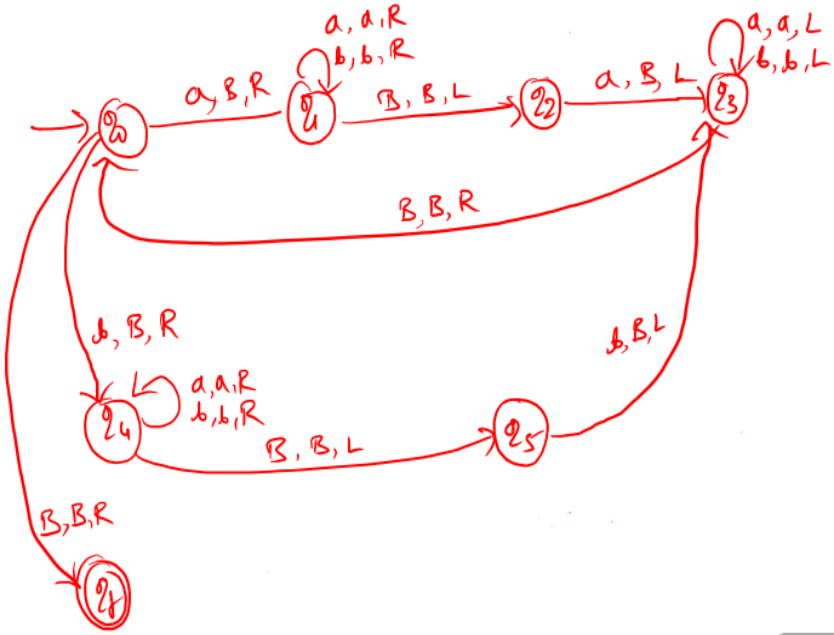
These two transitions are for accepting the odd length palindromes, and shall be excluded if the problem is for accepting **ww**^{r}

$(q_2, B) \rightarrow (q_f, B, R)$
$(q_5, B) \rightarrow (q_f, B, R)$
} a in center odd length Palindrome, b in center

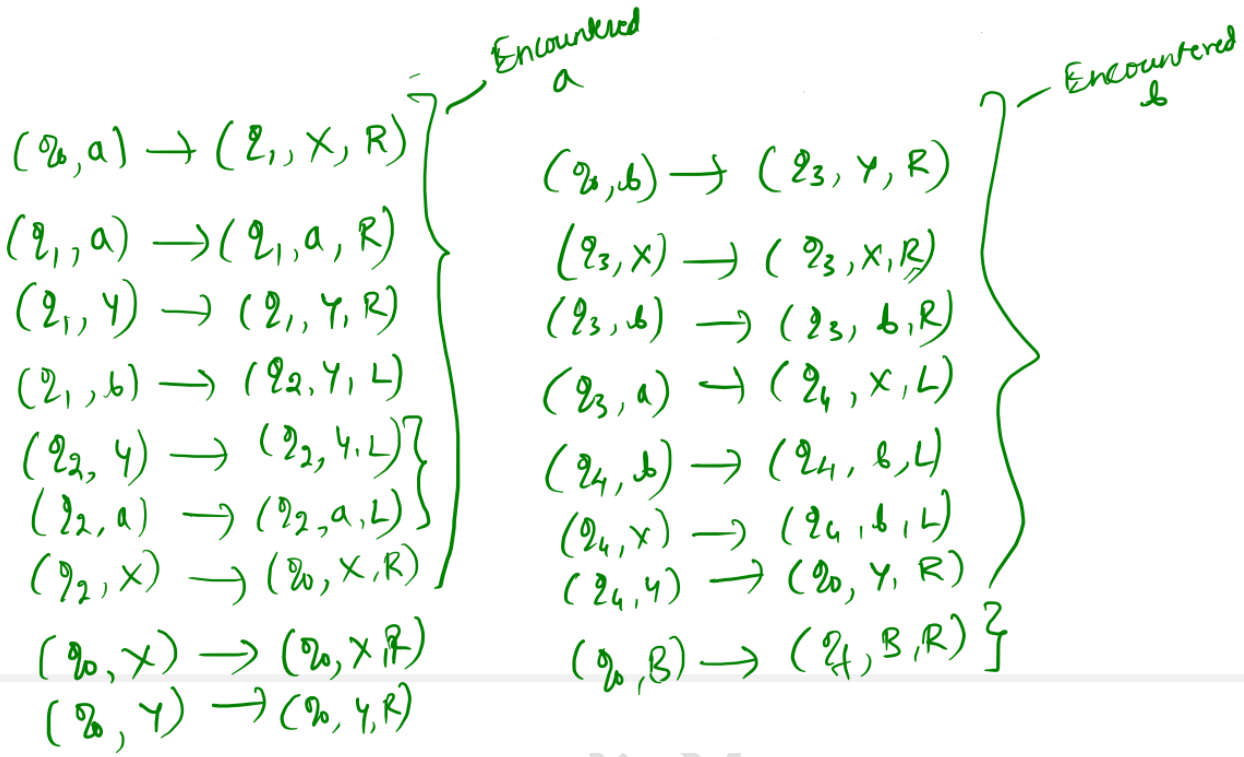
Problem 4: Design TM that accepts {ww^r| w ∈(a+b)*}

Note: The problem of Palindrome (problem 3) and ww^r (problem 4) is similar except for Palindrome includes the transition for both odd and even length palindromes, whereas ww^r includes only even length.





Problem 5: Obtain a TM to accept L={w | w ∈(a+b)*, $N_a(w)=N_b(w)$}



Note: Write the transition diagram based on above transitions

Turing machine works by converting an a to X and look for its matching B to convert that in to Y, and traverses back to find the next symbol. If the encountered symbol is b then it converts the b to Y and looks for its matching a to convert in to X. This process continues till no further symbol left to be processed and hence halts in accepting state.

4.5 LANGUAGE ACCEPTABILITY BY TURING MACHINE

Let us consider the Turing machine M= (Q, Σ, Γ, δ, q0, b, F), A string w in Σ* is said to be accepted by M if

$$q_0w \vdash \alpha_1 p \alpha_2$$

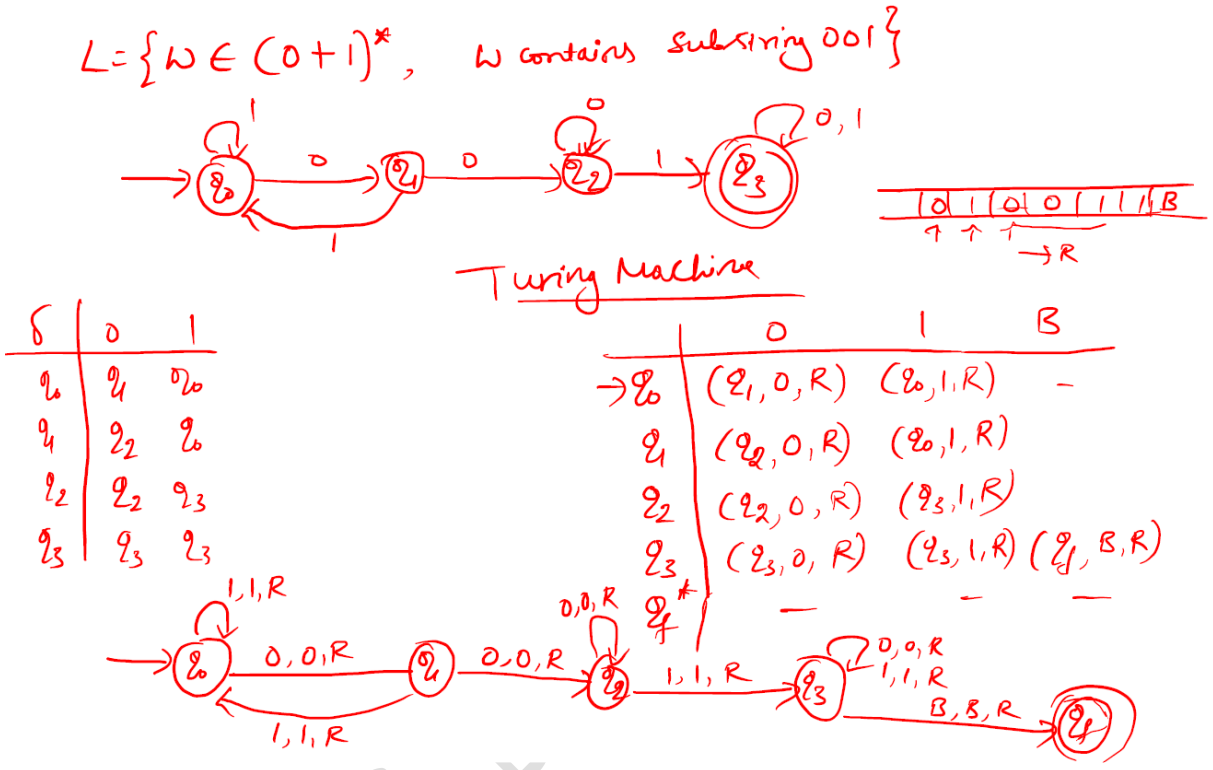
For some p ∈ F and α1, α2 ∈ Γ*

M does not accept w if the machine M either halts in a nonaccepting state or does not halt.

Problem 6: Obtain a TM to accept the string containing substring 001

Regular languages for which DFSM can be constructed initially and convert the DFSM to TM, TM thus constructed will have its R/W head always moving to Right on each transition.

For Ex:



Each transition of DFSM of the form (qi,a)->(qj) will take TM transition of the form

(qi,a)|-(qj,a,R)

And one addition transition is added from each final state of DFSM (qi,B)|-(qf,B,R)

4.6 BASIC GUIDELINES FOR DESIGNING TURING MACHINE

Basic guidelines for designing a Turing machine.

I. The fundamental objective in scanning a symbol by the R/W head is to 'know' what to do in the future. The machine must remember the past symbols scanned. The Turing machine can remember this by going to the next unique state.

II. The number of states must be minimized. This can be achieved by changing the states only when there is a change in the written symbol or when there is a change in the movement of the R/W head.

4.7 TECHNIQUES FOR TM CONSTRUCTION

Techniques for TM construction:

1. **TURING MACHINE WITH STATIONARY HEAD**

In the definition of a TM we defined *δ(q, a)* as *(q', y, D)* where *D =L* or *R.* So 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 the same cell for some input symbol. Then we define *δ(q, a)* as *(q', y, S).*

Thus in this model *δ(q. a) =(q', y, D)* where *D =L, R or S*.

2. STORAGE IN THE STATE

We are using a state, whether it is of a FA 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

M = ({q0, q1 x {0, 1, b}, {0, 1}, {0, 1, b}, δ, [q0, b], {[q1,b]})

3. MULTIPLE TRACK TURING MACHINE

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.**

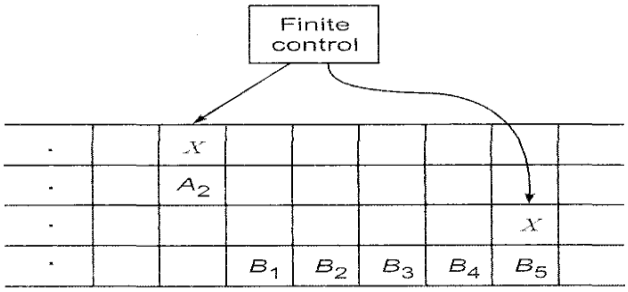


Fig: Multitrack TM

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 when the return state of the subroutine is reached.

Ex: TM to do multiplication of two positive integers.

4.8 VARIANTS OF TURING MACHINE

4.8.1 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

$$Q \times \Gamma^k \text{ into } (Q \times \Gamma^k \times \{L, R, S\}^k)$$

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:

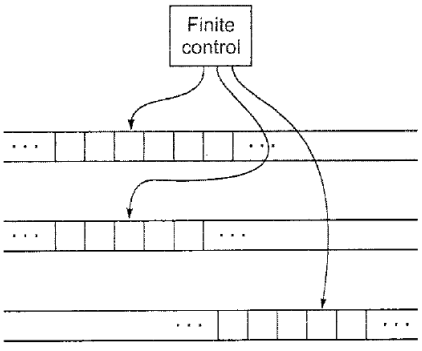


Fig. 9.8 Multitape Turing machine.

4.8.2 NONDETERMINISTIC TURING MACHINES

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 power set of (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.

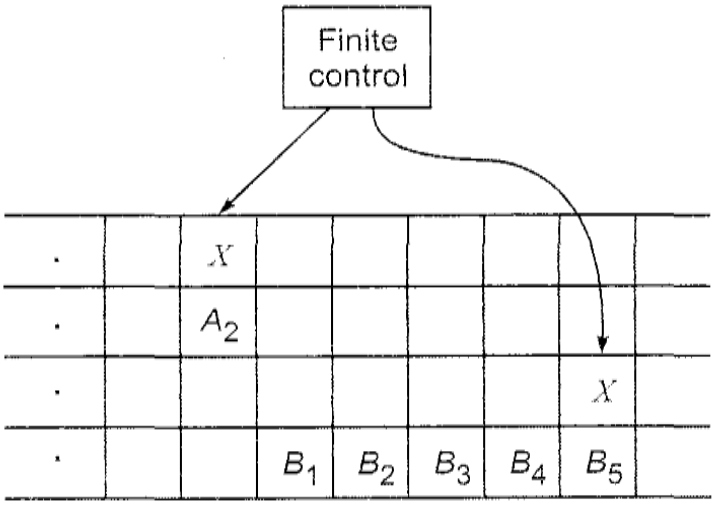
Theorem: Every language accepted by a multitape TM is acceptable by some single-tape TM (that is, the standard TM).

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 M with a singletape for the case k =2 is done in the fig. The construction can be extended to the general case.

The symbols A2 and B5 are the current symbols to be scanned and so the headmarker X is above the two symbols.

And hence the proof.

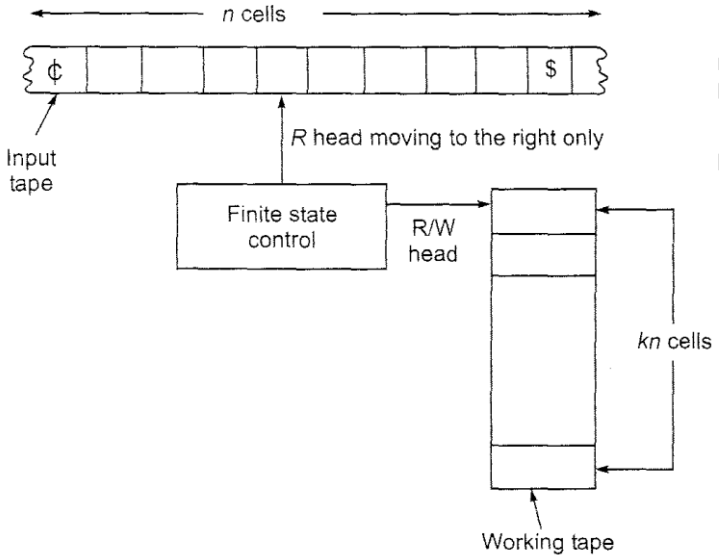


4.9 THE MODEL OF LINEAR BOUNDED AUTOMATON:

This model is important because

(a) the set of context-sensitive languages is accepted by the model. and

(b) the infinite storage is restricted in size but not in accessibility to the storage in comparison with the Turing machine model. It is called the linear bounded automaton (LBA) because a linear function is 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 LBA models can be described formally by the following set format:

- M = (Q, Σ, Γ, δ, q0, b, ¢, $, F)

Input alphabet L contains two special symbols ¢ 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.

QUESTION BANK FOR MODULE-4

1. Define Turing machine and explain with neat diagram, the working of a Basic Turing machine. (08M -Sep 2020)
2. Design a TM to accept the language $\{0^n1^n2^n \mid n>=1\}$ and show the moves made by the machine for 000111222. (hint: same as $a^n b^n c^n$ } (10M Sep2020)
3. Briefly explain the techniques for Turing machine construction. (05M Sep2020)
4. Explain the following terms
    a. Non-Deterministic Turing Machine
    b. Multitape Turning Machine (10M Sep 2020)
5. Explain Multitape Turing Machine with diagram. Prove that every language accepted by a Multitape TM is acceptable by some standard TM. (11M Dec 2019)
6. Explain the model of Linear Bounded Automata (05M Dec 2019)
7. Design a Turing machine to accept strings of a's and b's ending ab or ba. (08M Jan2019) (hint: Construct DFSM and then convert to TM)
8. Write a short note on various types of Turing machine (06M Jan2020)