# Formal definition of a Turing Machine:

A Turing machine is a 7-tuple, (Q , Σ ,  Γ , δ , q0 , q Accept , q Reject ), where Q , Σ ,  Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the special *blank* symbol ⌴,
3. Γ is the tape alphabet, where ⌴  ϵ Γ and Σ ⊆ Γ,
4. δ: Q x Γ → Q x Γ x { L , R } is the transition function,
5. q0 ϵ Q is the start state,
6. q Accept ϵ Q is the acceptance state, and
7. q Reject is the reject state, where q Reject ≠ q Accept

Often a few extra symbols are needed to make computation easier. ⌴ is a special symbol which allows the TM to know where its input ends by just searching for a blank.

# Multi-tape Turing machines:

1. A multi-tape Turing machine is like an ordinary Turing machine with several tapes.
2. Each tape has its own head for reading and writing.
3. Initially, the input is written on the first tape, and all the other tapes are blank, with each head at the beginning of the corresponding tape.
4. The transition function is changed to allow for reading, writing, and moving the heads on some or all of the tapes simultaneously.

Formally, the transition function of the Multi-tape Turing machine is represented as

δ: Q x Γk → Q x Γk x { L , R , S }k,

where k is the number of tapes. The expression

δ (q i , a1 , . . . , ak ) = (q j , b1 ,. . . , bk , L , R, . . . , L )

It means that, if the machine is in state q I , and heads 1 through k are reading symbols a1 through ak , the machine goes to state q j , writes symbols b1 throughbk , and directs each head to move left or right, or to stay put, as specified.

“Two machines are equivalent if they recognize the same language.” But equivalent does not mean that the two machines have the same speed, efficiency, runtime, feasible to program, steps of computation etc.

Multi-tape Turing machines appear to be more powerful than ordinary Turing machines, but they are no more powerful than single-tape Turing machines. But multi-tape Turing machines are useful because they are a bit easier to program and more useful for some tasks.

**Simulation of Multi-tape Turing machines into Single-tape Turing Machine:**

**THEOREM:** Every multi-tape Turing machine has an equivalent single-tape Turing machine.

**Proof:** We show how to convert a multi-tape TM M to an equivalent single-tape TM S. The key idea is to show how to simulate M with S.

Say that M has k tapes. Then S simulates the effect of k tapes by storing their information on its single tape. It uses the new symbol # as a delimiter to separate the contents of the different tapes. In addition to the contents of these tapes, S must keep track of the locations of the heads. It does so by writing a tape symbol with a dot above it to nark the place where the head on that would be. Think of these as “virtual” tapes and heads. The “dotted” tape symbols are simply new symbols that have been added to the tape alphabet. The following figure illustrates how one tape can be used to represent three tapes.

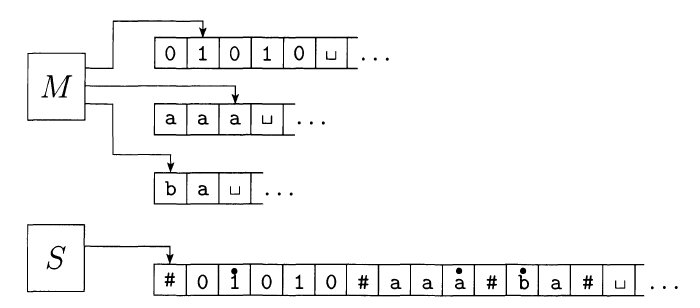


Fig 2.3Simulation of Multi-tape TM M to single-tape TM S.

For a 3-tape TM, a transition will look like

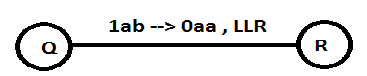


Fig 2.4 Transition in Multi-tape TM

If you are in state Q and see 1 on tape1, a on tape2 and b on tape3,

* Write 0 ontape1, a on tape2 and a on tape3
* Move head1 left, head2 left and head3 right
* Go to state R.”

S = “ On input w = w1 . . . wn :

1. First S puts its tape into the format that represents all k tapes of M. The formatted tape contains



1. To simulate a single move, S scans its tape from the first #, which marks the left-hand end, to the (k + 1)st #, which marks the right-hand end, in order to determine the symbols under the virtual heads. The S makes a second pass to update the tapes according to the way that M’s transition function dictates.
2. If at any point S moves one of the virtual heads to the right onto a ‘#’, the action signifies that M has moved the corresponding head onto the previously unread blank portion of that tape. So S writes a blank symbol on this tape cell and shifts the tape contents, from this cell until the rightmost #, one unit to the right. Then it continues the simulation as before.”

**REFERENCES:**

* I have taken help from chithkala basavaraj **project(Variants of Turing Machine)**  to understand the topic**.**
* Introduction to Theory of computation by Michael Sipser.