9.9.1. Multiple Tapes Turing Machine

One can think of turing machines that have several tapes (see Fig. 9.29). Each tape is connected to the finite control by means of a read/write head (one on each tape).

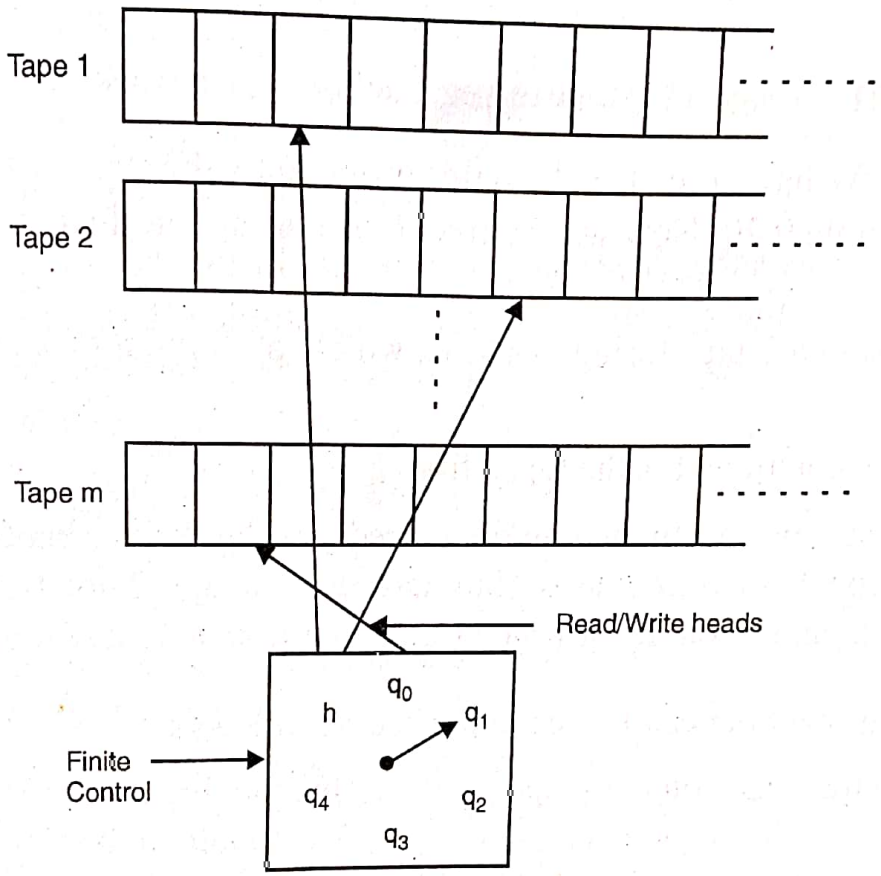

Fig. 9.30.

The machine can in one step read the symbols scanned by all its heads and then, depending on these symbols in current state, rewrite some of those scanned squares and move some of the heads to the left or right, in addition to changing the state.

For any fixed integer $m \geq 1$, an m-tape turing machine is a turing machine equipped with m tapes and corresponding heads. Thus standard turing machine studied so far in this chapter is just an m-tape turing machine, with $m = 1$.

Definition

Let $m \geq 1$ be an integer.

An m-tape turing machine is a six tuple machine as follows :

$$\text{Tm} = (Q, \Sigma, \Gamma, \delta, q_0, h)$$

where

Q : the finite set of states of the finite control

Σ : The finite set of input symbols

Γ : The tape symbol where $\Gamma = \Sigma \cup \#$

h : halt state $h \in Q$

q_0 : The initial state, $q_0 \in Q$

δ : The transition function is a function which maps :

$$(Q \times \Sigma^m) \rightarrow Q \times (\Sigma \cup \{L, R, N\}^m).$$

That is, for each state-q, and each m-tapes of tape symbols $(a_1, a_2, ..., a_k)$, $\delta(a_1, a_2, ..., a_k) = (P, (b_1, b_2 ...b_k))$, where P is, as before, the new state, and b_j is the action taken by Tm at tape j.

Computation takes place in all m-tapes of a m-tape Turing machine. Accordingly, a configuration of such a machine must include information about all tapes.

9.9.3. Multiple Heads Turing Machines

In order to simplify the discussion, we assume that there are only two Heads on the tape. The tape is assumed to be one-way infinite. We explain the involved concepts with the help of an example. Let the content of the tape and the position of the two heads that is H_1 and H_2, be as given below :

$$\#\#a\ \underset{\uparrow}{b}\ c\#d\ \underset{\uparrow}{e}\ f\#\#\ldots$$
$$H_2 \qquad H_1$$

Further, let the state of the Tm be q.

The move function of the two-head one-way turing may be defined as

δ(state, symbol under Head1, symbol under Head2)

$$= \text{(New state, } (S_1, M_1), (S_2, M_2))$$

where S_i is the symbol to be written in the cell under H_i (the ith Head) and M_i denotes the movement of H_i, where the movement may be L, R or N and further L denotes movement to the left, R denotes movement to the right of the current cell and N denotes 'no movement of the Head'.

Two special cases of the δ function defined above, need to be considered :

(*i*) What should be written in the current cell when both Heads are scanning the same cell at a particular time and the next moves (S_1, M_1), (S_2, M_2) for the two heads, are such that $S_1 \neq S_2$ that is symbol to be written in current cell by H_1 is not same as symbol to be written in current cell by H_2.

In such a situation, a general rule may be defined, say, as whatever is to be done by H_1 will take precedence over whatever is to be done by H_2.

9.9.6. Recursively Enumerable and Recursive Languages

When a Turing machine executes an input, there are four possible outcomes of execution. Then Tm

(*i*) Halts and accept the input

(*ii*) Halts and rejects the input

(*iii*) Never halts (fall into loop), or

(*iv*) Crash.

We also know that the language accepted by Tm is known as recursively enumerable and the set of recursive languages is subset of recursive enumerable set.

The set of recursively enumerable languages are precisely those languages that can be listed (enumerated) by a turing machine. Basically, if a Tm can generate all the strings in a language, another Tm can accept all the strings that are not in the given language. (Strings that are not in the language may be rejected or may cause the turing machine to go into an infinite loop).

TIPS

A language is recursive if there exists a Turing machine that accepts every string of the language and rejects every string that is not in the language. So, we are sure about the rejection of the strings with are not in the given recursive language.

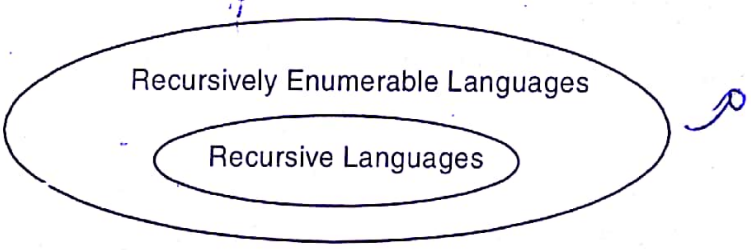

Fig. 9.34.

Theorem 9.2. If a language L is recursive, then it is recursively enumerable languages.

Proof. A language is recursively enumerable if there exists a Turing machine that accepts every string of the language and a language is recursive if there is a Tm that accepts every string of the language and does not accept strings that are not in the language.

So, every recursive language is also recursively enumerable. Hence, statement of the theorem is true.

9.9.7. Turing Machine as Enumerators