6.6 VARIANTS OF TURING MACHINE

There are number of other types of Turing machines in addition to the one we have seen. To make the Turing machines more powerful we could add to its features :

- Let its tape extend indefinitely in both directions.
- Let its tape have multiple tracks.
- Let there be several tapes, each with its independent tape head.
- Add non-determinism.

It turns out that computationally all these Turing machines are equally powerful. That is, what one type can compute any other can also compute. However, the efficiency of computation, that is, how they can compute, may vary.

6.6.1 Turing Machine with Infinite Tape

This is a kind of Turing machine that have one finite control and one tape which extends indefinitely in both directions. It turns out that this types of Turing machines are only as powerful as one tape Turing machines whose tape has a left end.

6.6.2 Multiple Turing Machines

A multiple Turing machine has 'k' tapes. It takes its input on tape 1, other tapes are initially blank. The transition function now has type:

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$$

It specifies how the k-tape heads behave when the machine is in state q_i, reading $a_1, \ldots, a_k$:

$$\delta(q_i, a_1, \ldots, a_k) = (q_i, (b_1, \ldots, b_k), (d_1, \ldots, d_k))$$

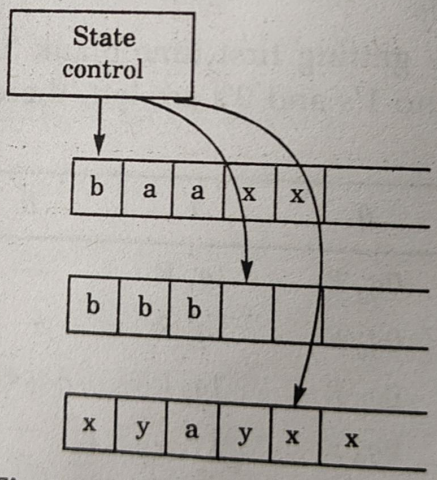

Fig. 6.6.1. Multitape Turing Machine.

Theorem. A language is ***Turing recognizable*** if some multiple Turing machine recognizes it.

Proof : Let as see how to simulate a multitape Turing M by a standard TM S. The standard machine has tape alphabet $\{\#\} \cup \Gamma \cup \Gamma'$ where # is a separator, not in $\Gamma \cup \Gamma'$, where there is one-to-one correspondence between symbols in Γ' and (marked) symbols in Σ'.

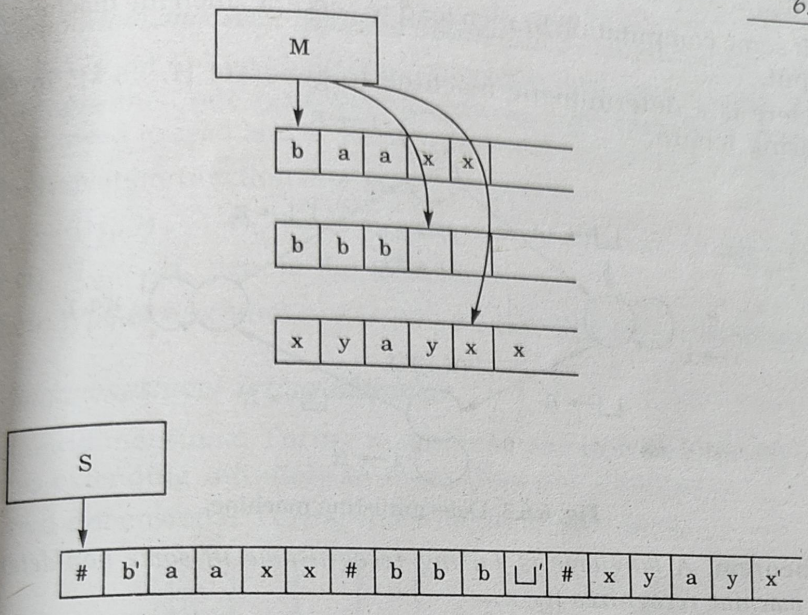

Fig. 6.6.2. Multitape turing machine simulated in a single tape.

S reorganizes its input $x_1, x_2, ..., x_n$ into

$$\# x_1' x_2 ..... x_n \# \underbrace{\sqcup' \# ... \# \sqcup' \#}_{\text{K-1 times}}$$

Note : How symbols of Γ' represent marked symbols from Γ, which denote the positions of the tape heads in the multitape Turing machine.

To simultate a move of M, S scans its tape to determine the marked symbols. S then scans the tape again, updating it according to M's transition function.

If a "virtual head" of M moves to a #, S shifts that symbols and every symbol after it, one cell to the right. In the vacant cell it writes ⊔. It then continues to apply M's transition function.

6.6.3 Non-deterministic Turing Machines

A non-deterministic Turing machine is a Turing machine, which is like non-deterministic finite automatas, at any state it is in and for 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. Here an action means the combination of writing a symbol on the tape, moving the tape head and going to a next state.

A non-deterministic Turing machine has a transition function of type :

$$\delta : Q \times \Gamma = P\,(Q \times \Gamma \times \{L, R\})$$

If some computation branch lead to "accept" then the machine accepts its input.

Here is a deterministic machine to generate $\{1, ..., k\}^*$ in order of increasing length.

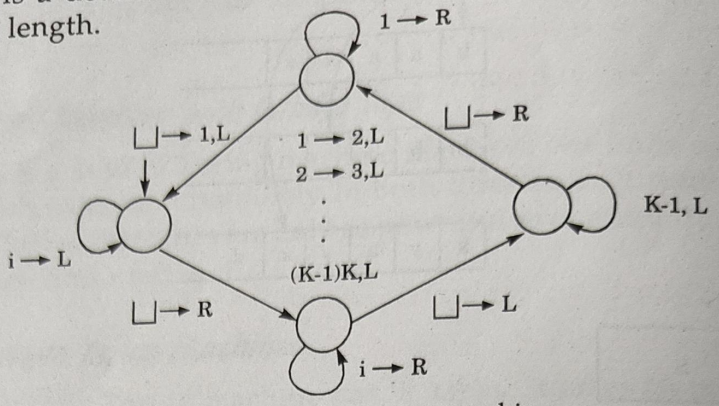

Fig. 6.6.3. Deterministing machine.

Theorem. *A language is Turing recognisable iff some non-deterministic Turing machine recognizes it.*

Proof : We need to show that every non-deterministic Turing machine M can be simulated by a deterministic Turing machine D.

Let us see how it can be simulated by a 3-tape machine.

Let k be the largest number of choices, according to N's transition function, for any state/symbol combination.

Tape 1 contains the input.

Tape 3 holds progressively longer and longer sequences from $\{1, ..., k\}^*$.

Tape 2 is used to simulate N's behaviours for each fixed sequence of choices given by tape 3.

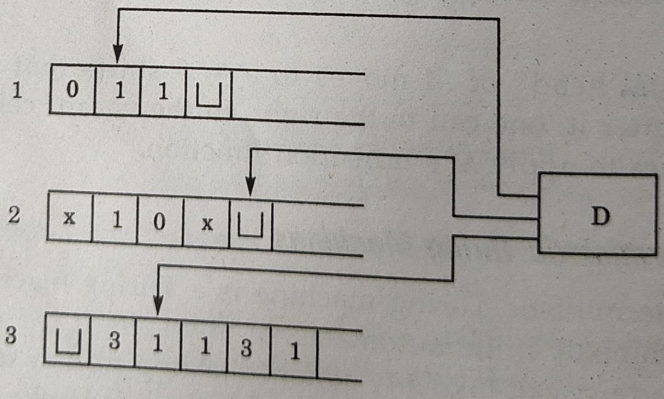

Fig. 6.6.4

1. Initially tape 1 contains input w. The other two tapes are empty.
2. Overwrite tape 2 by w.
3. Use tape 2 to simulate N. Tape 3 dictates how N should make its choices. If tape 3 gets exhausted, go to step 4. If N says accept, accept.
4. Generate the next "choice" string on tape-3. Go to step 2.

program and carry out the operations called for without the exercise of any ingenuity or insight. If Turing's thesis is correct, then talk about the existence and non-existence of effective methods can be replaced throughout mathematics and logic by talk about the existence or non-existence of Turing machine programs.

Turing introduced his thesis in the course of arguing that the Entsecheidungs problem, or decision problem, for the predicate calculus-possed by Hilbert (Hilbert and Ackermann 1928)—is unsolvable. Here is church's account of the Entscheidungs problem.

"By the Entscheidungs problem of a system of symbolic logic is here understood the problem to find an effective method by which, given any expression Q in the notation of the system, it can be determined whether or not Q is provable in the system."

The truth table test is such a method for the propositional calculas. Turing showed that, given his thesis, there can be no such method for the predicate calculus. He proved formally that there is no Turing machine which can determine, in a finite number of steps, whether or not any given formula of the predicate calculus is a theorem of the calculus. So, given his thesis that if an effective method exists then it can be carried out by one of his machines, it follows that there is no such method to be found.

Intactive notion of algorithms	equal	Turing machine algorithms

Fig. 6.7.1. The church turing thesis.

6.8 UNIVERSAL TURING MACHINE

A Turing machine as we have already studied is the mathematical tool equivalent to a digital computer. It was suggested by the mathematician Turing in the 30s, and has been since the most widely used model of computation in computability and complexity theory.

The model consists of an input output relation that the machine computes. The input is given in binary form on the machine's tape, and the output consists of the contents of the tape when the machine halts.

What determines how the content of the tape change is a finite state machine inside the Turing Machine. The FSM is determined by the number of states it has, and the transitions between them.

At every step, the current state and the character read on the tape determine the next state the FSM will be in, the characters that the machine will output the tape (possibly the one read, leaving the contents unchanged), and which direction the head moves in, left or right.

This is why the notion of a Universal Turing Machine (UTM), had been introduced, which along with the input on the tape, takes in the description

of a machine M. The UTM can go on then to simulate M on the rest of the contents of the input tape. A universal Turing machine can thus simulate any other machine.

UTM can be defined as a Turing Machine that is capable of simulating any other Turing Machine by encoding the latter.

6.9 LINEAR BOUNDED AUTOMATA

A Turing machine has an infinite supply of a blank tape. A **linear-bounded automata** (*LBA*) is a Turing machine whose tape is only kn squares long, where 'n' is the length of the input (initial) string and k is a constant associated with the particular linear-bounded automaton.

This model is important because :

- the set of context-sensitive languages is accepted by the model and
- the infinite storage is restricted in size but not in accessibility to the storage in comparison with the Turing machine model. It is called linear bounded automata (LBA) because a linear function is used to restrict (to bound) the length of the tape.

A linear bounded automaton is a non-deterministic Turing machine which has the 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 some meaning as in the basic model of Turning machines with the differences that the input alphabet Σ contains two special symbols $¢$ and $\$$.

$¢$: called left-end marker, which is entered in the left-most cell of the input tape and prevents the R/W head from getting off the left end of the tape.

$\$$: called right-end marker which is entered in the right-most 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 R/W head should not print any other symbol over both the end markers.

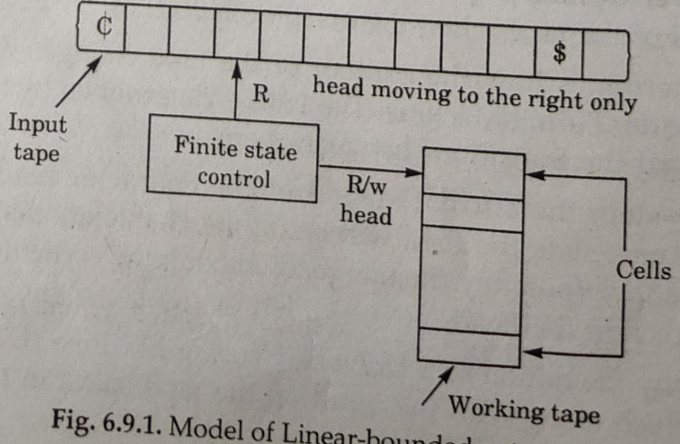

Fig. 6.9.1. Model of Linear-bounded automata.