Reviewed Paper | Volume 2 | Issue 2 | October 2014

International Journal of Informative & Futuristic Research | ISSN (Online): 2347-1697

Technical framework of Operating System using Turing Machines

Paper ID	IJIFR/ V2/ E2/ 028	Page No	465 - 470	Subject Area	Computer Science
Key Words	Turing, Undesirability, Complexity, Snapshot				

Dr. Ruchi Gupta	Assistant Professor Department Of Computer Science Government P.G. College, Ambala Cantt, Haryana

Abstract

Turing thesis states that any algorithm procedure that can be carried by human beings/computer can be carried out by a Turing Machine. Turing Machines are also used for determining the undesirability of certain languages and measuring the space and time complexity of problems. 'Snapshots' of a Turing machine in action can be used to describe a Turing machine. The machine must remember the past symbol scanned. The Turing machine can remember this by going to the next unique state. In a multiple track TM, a single tape is assumed to be divided into several tracks. We know subroutines are used in computer languages, when some task has to be done repeatedly. This facility can implement for Turing Machines. Turing machines are useful in several ways. As an automaton, the Turing machine is the most general model. The Turing machine can be thought of as finite control connected to a R/W (read/write) head. A Turing machine computes a function $f : \Sigma^* \rightarrow \Sigma^*$ *if, for any input word w, it always stops in configuration where f(w) is on the tape. The functions that are computable by an effective procedure are those that are computable by a Turing Machine.*

1. Introduction

Kleene in 1935, Schonfinkel in 1965 gave various models using the concept of Turing machines, λ-calculus, combinatory logic, post-system and μ-recursive functions. It is interesting to note that these were formulated much before the electro-mechanical/electronic computer were devised. Although these formalisms, describing effective computation, are dissimilar, they turn to be equivalent. Among these formalisms, the Turing's formulation is accepted as a model of algorithm or computation. It has been accepted by computer scientists that the Turing machine provides an ideal theoretical model of a computer. A Turing machine can both write in the tape and read from it, the read-write head can move both to left and right. The tape is infinite. Once accept/reject states ate reached, the computation terminates at once.

ISSN (Online): 2347-1697
International Journal of Informative & Futuristic Research (IJIFR)
Volume 2, Issue 2, October 2014
14th Edition, Page No: 465-470

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 machine proposed his 'Computing Machine'.

A Turing machine M is a 7-tuple, namely (Q, ∑, Ґ, δ, q0, b, F) where

1. Q is a finite nonempty set of states,
2. Ґ is finite nonempty set of tape symbols,
3. b ∈ Ґ is the blank,
4. ∑ is a nonempty set of input symbols and is a subset of Ґ and b ∉ ∑,
5. 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.
6. q0 ∈ Q is the initial state, and
7. F ⊆Q is the set of final states.

2. Model of Turing Machine

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. The block diagram of the basic model for the Turing machine is given in Fig. 1.

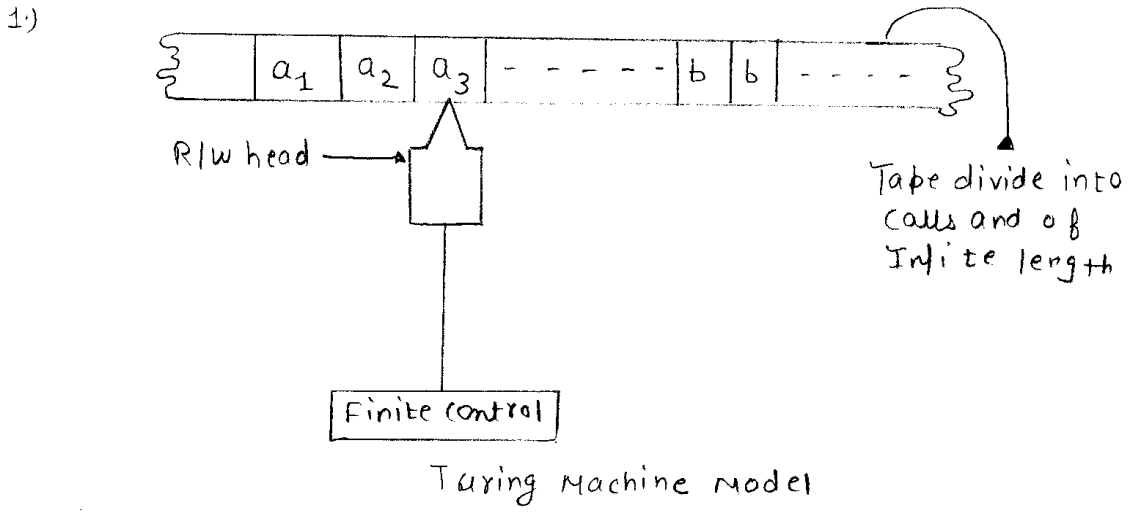

Figure 1:Turing machine Model

Each cell can store only one symbol. The input to and the output from the finite state automaton are effected 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 the present state of an automaton to determine.

3. Representation of Turing Machine By Instantaneous Description

'Snapshot' of a Turing machine in action can be used to describe a Turing machine. These give 'Instantaneous Descriptions' of Turing Machine. We have define instantaneous description of a PDA (Push-Down automaton) in terms of the current state, the input string to be processed, and the topmost symbol of the pushdown store. But the input string to be processed is not sufficient to be defined as the ID of a Turing machine, for the R/W head can move to the left as well. So an ID of a Turing Machine is defined in terms of the entire string and the current state.

Example: *A snapshot of Turing machine is shown in Fig-2. Obtain the instantaneous description.*

ISSN (Online): 2347-1697
International Journal of Informative & Futuristic Research (IJIFR)
Volume 2, Issue 2, October 2014
14th Edition, Page No: 465-470

***Solution*:** The present symbol under the R/W head is a_1. The present state is q_3. So a_1 is written to the right of q_3. The nonblank symbols to the left of q_1 form the string $a_4a_1a_2a_1a_2a_2$, which is written to the left of q_3. The sequence of nonblank symbols to the right of a_1 is a_4a_2. Thus the ID is as given in Fig-3.

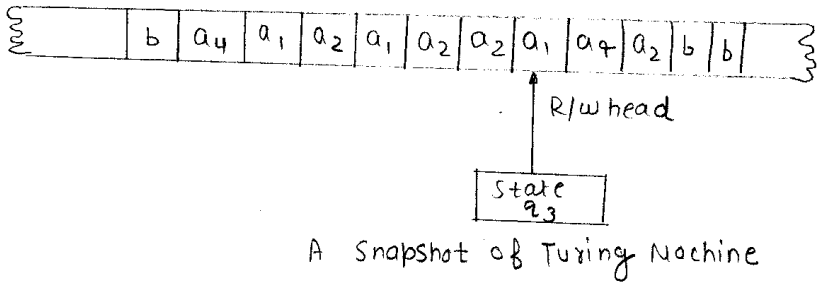

Figure 2: A snapshot of Turing Machine

For constructing the ID, we simply insert the current state in the input string to the left of the symbol under the R/W head. We observe that the blank symbol may occur as part of the left or right substring.

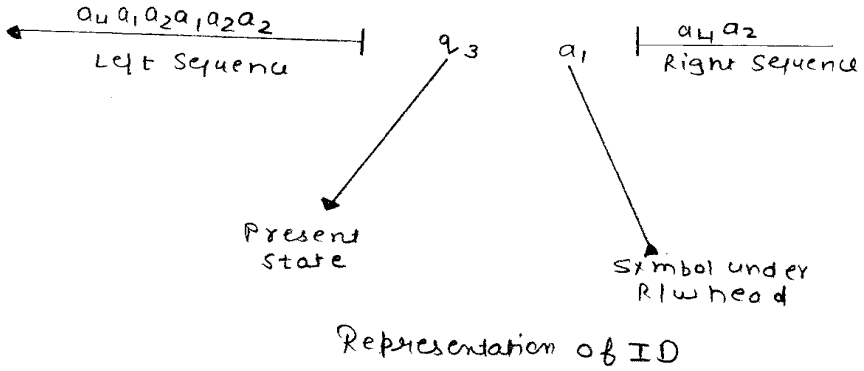

Figure 3: Representation of ID

4. Variants of Turing Machines

The Turing machine we have introduced has a single tape. δ(q, a) is either a single triple (p, y, D), where D = R or L, or is not defined. We introduce two new models of TM :

(i) A TM with more than one tape

(ii) A TM where $\delta(q, a) = \{ (p_1, y_1, D_1), (p_2, y_2, D_2), \ldots, (p_r, y_r, D_r) \}$. The first model is called a multitape TM and the second a nondeterministic TM.

4.1 Multitape Turing Machines

A multitape TM has a finite set Q of states, an initial state q_0, 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. 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 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 $Q \times \acute{\Gamma}^k$ into $Q \times \acute{\Gamma}^k \times \{L, R, S\}^k$. we use implementation description to define δ. Figure-4 represent a multitape TM. A move depends on the current state and k tape symbol under k tape heads.

ISSN (Online): 2347-1697
International Journal of Informative & Futuristic Research (IJIFR)
Volume 2, Issue 2, October 2014
14th Edition, Page No: 465-470

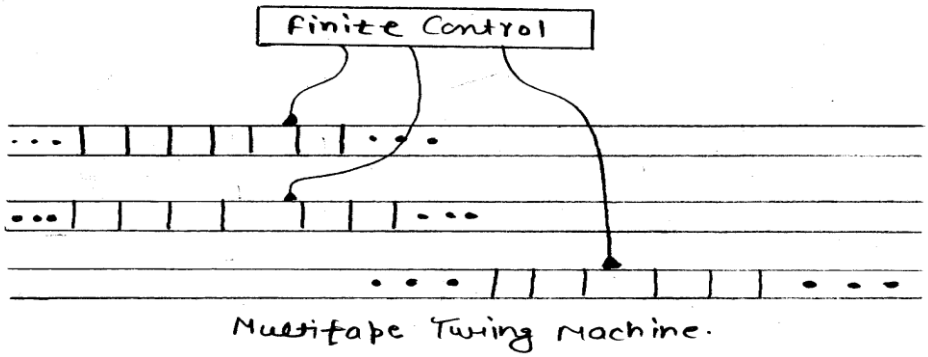

Figure 4: Multitape Turing machine

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 q_0, the input string w in the first tape, empty strings of b's in the remaining k-1 tapes. An accepting ID has a final state, some strings in each of the k tapes.

Figure-5 can be used to visualize the simulation. The symbols A_2 and B_5 are the currents symbols to be scanned and so the head marker X is above the two symbols.

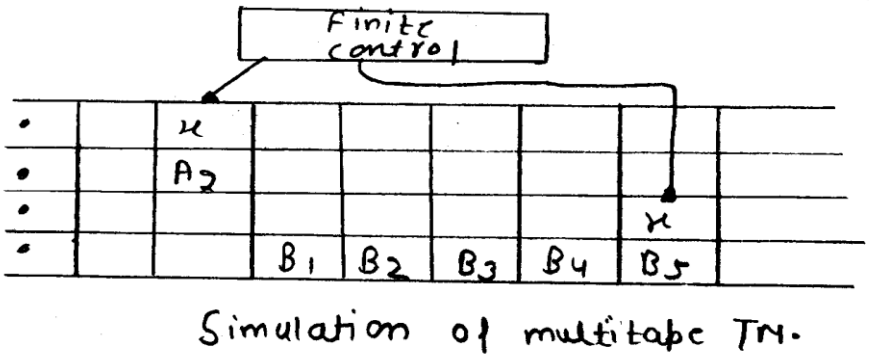

Figure 5: Simulation of Multitape Turing machine

Initially the contents of tapes 1 and 2 of m are stored in the second and fourth tracks of M_1. 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 M_1 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 M_1. Note that the finite control of M_1 has also the information about the states of M and its moves. After visiting both head markers, M_1 knows the tape symbols being scanned by the two heads of M.

Now M_1 revisit each of the head markers:

(i) It changes the tape symbol in the corresponding track of M_1 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) M_1 changes the state of M in its control.

ISSN (Online): 2347-1697
International Journal of Informative & Futuristic Research (IJIFR)
Volume 2, Issue 2, October 2014
14th Edition, Page No: 465-470

This is the simulation of a single move of M. At the end of this, M_1 is ready to implement its next move based on the revised positions of its head markers and the changed state available in its control. M_1 accepts a string w if the new state of M, as recorded in its control at the end of the processing of w, is a final state of M.

4.2 Nondeterministic Turing Machines

In the case of standard Turing machines (hereafter we refer to this machine as deterministic TM), $\delta(q_1, a)$ was defined (for some elements of Q X Ґ) as an element of Q X Ґ X {L, R}. Now we extend the definition of δ. In a nondeterministic TM, $\delta(q_1, a)$ is defined as a subset of Q X Ґ X {L, R}.

5 Objectives of the study

A Turing Machine can both write on the tape and read from it and read-write head can move both to the left and to the right. Turing machine have an unlimited amount of storage space for their computations. Turing machine are not intended to model computers, but rather they are intended to model computation itself.

The difference lies only with the ability of a Turing machine to manipulate an unbounded amount of date. However, given a finite amount of time, a Turing machine (like a real machine) can only manipulate a finite amount of data. Like a Turing machine, a real machine can have its storage space enlarged as needed, by acquiring more disks or other storage media. If the supply of these runs short, the Turing machine may become less useful as model. But the fact is that neither Turing machines nor real machines need astronomical amounts of storage space in order to perform useful computation. The processing time required is usually much more of a problem.

6 Conclusions

Turing machines describe algorithms independent of how much memory they use. There is limit to the memory possessed by any current machine, but this limit can rise arbitrarily in time. Turing machines allow us to make statements about algorithms which will hold forever, regardless of advances in conventional computing machine architecture. Turing machines are that they do not model concurrency well. For example, there is a bound on the size of integer that can be computed by an always-halting nondeterministic Turing machine starting on blank tape. The Turing machine arose as an ideal theoretical model for an algorithm. The Turing Machine provides machinery to mathematicians for attacking the Hilberts' tenth problem. The problem can be restated as follows: does there exist a TM that can accept a polynomial over n variables if it has an integral root and reject the polynomial if it does not have one. Using Gödel numbering which converts operations of Turing machines into numeric quantities, it can be proved that Turing-computable functions are partial recursive.

References

[1] Kohavi, ZVI, Switching and Finite Automata Theory, Tata McGraw-Hill, New Delhi, 1986.
[2] Sahni, D.F. and D.F. McAllister, Discrete Mathematics in Computer Science, Prentice-Hall, Enhlewood Cliffs, New Jersy, 1977.
[3] Ullman, J.D., Fundamental Concepts of Programming Systems, Addison-Wesley, Reading (Mass.), 1976.
[4] Nelson, R.J., Introduction to Automata, Wiley, New York, 1968.
[5] Manna, Z., Mathematical Theory of Computation, McGraw-Hill, Kogakusha, Tokyo, 1974.

ISSN (Online): 2347-1697
International Journal of Informative & Futuristic Research (IJIFR)
Volume 2, Issue 2, October 2014
14th Edition, Page No: 465-470

[6] Levy, L.S., Discrete Structures of Computer Science, Wiley Eastern, New Delhi, 1988.
[7] Gries, D., The Science of Programming, Narosa Publishing House, New Delhi, 1981.
[8] Hopcroft, J.E., J. Motwani, and J.D. Ullman, Introduction to Automata Theory, Languages and Computation, Pearson Education, Asia, 2002