Enigmatic Power of Turing Machines: A Review

Amandeep Kaur
M.Tech Scholar, Central University of Punjab
Bathinda, India

Abstract— Automata are said to be theoretical devices which help in understanding the reasoning behind computations and are considered to be ideal rather than realistic. This is because the actual computers cannot have infinite memory as assumed in some automata. The Turing machine is a simple yet powerful mathematical model which is used for accepting and translating languages. It can also be used for evaluating functions. In this paper, the basic terminology related with Turing machine, which accounts for its high computing power, its uses in various areas, its variants, its relation with Artificial Intelligence has been discussed.

Keywords-Automata, Turing Machine, Artificial Intelligence

I. INTRODUCTION

Automata have been playing a continuously increasing, and have by now attained a considerable, role in natural sciences. This process is going on for several decades. During the last part of this period, Automata have begun to invade some parts of Mathematics, too particularly, but not exclusively, Mathematical Physics or Applied Mathematics [1]. Named after Alan M. Turing (1912-1954), the brilliant English mathematician and philosopher whose seminal paper in 1936 marked the beginning of the theory of computation, Turing Machine accounts to be a theoretical model having Random Access Memory. Turing also pioneered the fields of artificial intelligence and chess-playing by computer, as well as that of morphogenesis in biology, and was instrumental in breaking "Enigma", the German naval code during World War II [2]. Turing Machine is a powerful Automata having a Read and Write memory which allows it to access the symbols from the Input Alphabet (∑) in any arbitrary order, where Input Alphabet is a set consisting of the input symbols. Turing machines seem to form a stable and maximal class of computational devices, in terms of the computations they can perform. Turing machines are designed to satisfy simultaneously three criteria:

- They should be automata; that is, their construction and function should be in the same general spirit as for automata.
- They should be as simple as possible to describe, define formally, and reason about.
- They should be as general as possible in terms of the computations they can carry out [3].

II. STRUCTURE OF A TURING MACHINE

A Turing machine consists of an infinitely long tape divided into individual cells, a movable "head" to read and write characters on the cells, and a program that dictates how the head should react to the computations made by the machine. The tape extends infinitely to the right, with each successive cell containing exactly one symbol from the infinite alphabet $\sum = (a1; a2; .....an; b)$, where b is called a blank symbol. The Turing machine program invokes a infinite collection of Turing states $Q = (q0; q1;......; qk; qh)$, where qh denotes the distinguished halting state.

In essence, a Turing machine consists of a finite state control unit and a tape. Communication between the tape and the finite control unit is provided by a single head, which is used to read the symbols from the tape and can also change those symbols, which means it can Read as well as Write. The head can move in both the directions- Left as well as Right unlike other Automaton in which the Head was able to move only in one direction.

At each step, a Turing machine can perform two functions, which depend on its current state and the current symbol under the Read and Write Head.

- Step (i): Put the Control Unit in a new state.
- Step (ii): Either: Write a symbol in the cell of the tape under Read-Write Head replacing the one already under the head. Or: Move the Read-Write Head one cell of the tape to the left or right .

The input for the Turing machine is represented by the prepared initial pattern on the tape, and the output of the machine is given by the existing symbols on the tape when the machine reaches the halting state qh [4].

A. Formal Definition

A Turing machine is a quintuple $(Q, \sum, \delta, q0, H)$, where

- *K* is a finite set of states;
- ∑ is an alphabet, containing the blank symbol *b*, but not containing the symbols *L* and *R*.
- *q0* ε *K* is the initial state;
- *H* is the set of halting states and is a proper subset of *Q*.

δ, the transition function, is a function from $(Q \setminus \{qh\}) \times A \rightarrow (Q \setminus \{q0\}) \times A \times \{L; N; R\}$. This function takes as input, the present state which is not the Halting state and current symbol under the Head, and outputs the next state which is not the initial state, new symbol to be written and movement of the head to the Left (L) or Right (R). When it is not accepting anything it is said to be Not Moving (N).

B. Acceptance of a String by Turing Machine:

A string α is said to be accepted by a Turing machine if, when started in the initial state with the head positioned on the left-most character of the input string and the tape otherwise empty, the Turing machine halts and the state at that time is an accepting state. A language is recognized by a Turing machine if it is the set of all words accepted by the machine.

III. TERMINOLOGY RELATED TO TURING MACHINE

The basic terms and definitions related with the Turing machine have been discussed under the following heads.

A. Turing Recognizable or Recursively Enumerable Language

A language is Turing-recognizable or Recursively Enumerable if there is a Turing machine which recognizes it. A language is Turing-decidable or Recursive if there is a Turing machine which recognizes it which halts for all inputs.

B. Relation between Turing Decidable and Turing recognizable languages

Turing Decidable languages can be more formally defined by a statement that there exists a TM M such that for every string w, M running on input *α* halts and either rejects or accepts the string. And for Turing Recognizable languages there exists a TM M such that for every string w, M running on input *α* either halts and accepts or rejects *α*, or runs forever. As shown in Figure 1, Turing Decidable languages are a proper subset of Turing recognizable languages.

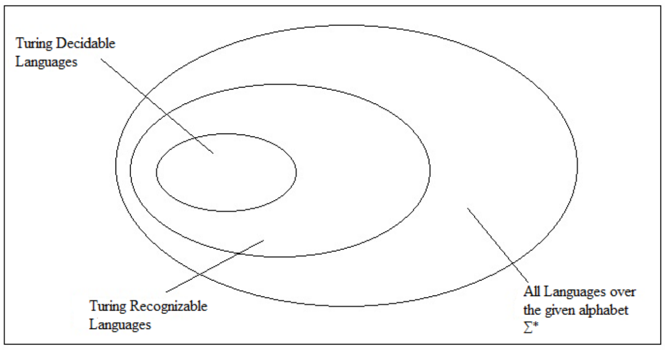
Turing Decidable Languages
Turing Recognizable Languages
All Languages over the given alphabet Σ*

Figure 1: Turing Decidable Languages

C. Turing Machines and Halting

One way for a Turing machine to accept input is to end in a final state. Another way is acceptance by Halting. A Turing machine is said to halt if it enters a state *q*, scanning a tape symbol *X, and* there is no move in this situation. That is *δ(q,X)* is undefined. In general, it is assumed that a Turing Machine always halts when it is in accepting state.

D. Recursive Languages

It is not always possible to require that a TM halts even if it does not accept the input. Turing machines that always halt, regardless of accepting or not accepting, are good models of algorithms for decidable problems. Such languages are called Recursive.

IV. AREAS WHERE TURING MACHINES ARE USED

A short introduction to the areas or subjects, where Turing Machines are used, is given in this section. More specifically, these are used in either Computation Theory or in Complexity Theory.

A. Computation Theory

This subject is concerned with the computability of functions and tells whether what can be computed by the Turing Machine and what cannot be. Various concepts covered under the computation theory where Turing Machines are used, are discussed below:

1) *Universal Turing Machine:* These are the machines which are said to simulate the work of any other Turing Machine. Universal Turing Machines are the central ingredient when non-computable functions are studied. Most common way to construct a Universal Turing machine is to use a multiple-tape Turing Machine. Then it can be shown that such a Turing Machine can be simulated by a single-tape Turing Machine.
2) *A Busy Beaver:* A *Busy Beaver* (BB) is said to be a binary Turing Machine which writes a whole lot of 1's on the cells of the tape before halting on a particular input. An open contest is going on, where the goal is to construct a Busy Beaver with less than some *n* amount of states and produces the largest amount of 1's.
3) *Boolean Circuits:* Given a Turing machine, it is possible to construct a Boolean circuit which can simulate all its computations if just we put a bound on the Turing machines memory (i.e., the number of cells in the tape is finite).

B. Complexity Theory

Complexity is usually defined as the number of steps or memory taken by a particular function to compute results. Various concepts covered under the complexity theory where Turing Machines are used, are discussed below:

1) *Unsolved Problem in Complexity Theory:* A very famous and at the moment unsolved problem in complexity theory is the whether P=NP? problem. P is the class of decision problems which can be solved by a Turing machine in polynomial time, and NP is the class of decision problems which can be solved by a non-deterministic Turing machine in polynomial time. The fact that if one can find just one problem in the class NP-complete, which is a special subset of NP which is also in P, then all of NP is inside P makes it an essential tool for studying P=NP?
2) *Probabilistic Turing Machine:* A non-deterministic Turing machine which chooses between its possible transitions in a random way to solve a problem is called a Probabilistic Turing Machine. For each possible transition there is an associated probability and therefore, the end result of the Turing machine will have some probability of being correct. By running the algorithm several times it is possible to minimize the chance of the result being incorrect.
3) *Non Determinism:* It seems that non-determinism makes an algorithm more powerful when viewing its time usage. But when viewing space usage it is known that PSPACE=NPSPACE. Non-deterministic Turing machines which use polynomial space can be simulated by deterministic Turing machines which also only use polynomial space [6].

V. TURING MACHINE VARIANTS

There are many variations that can be made to a basic Turing Machine, two of which have been discussed under the following heads:

A. Multitape Turing Machines

A Multitape Turing Machine is like an ordinary Turing Machine but it has several tapes instead of one tape. Initially the input starts on first tape and other tapes are left blank. The transition function is allowed to to read, write and move the heads on all the tapes simultaneously. This means that it is possible to read and write on multiple tapes and move in different directions. A Multitape Turing Machine is equivalent in power to an ordinary Turing Machine. Two Turing Machines are said to be equivalent if they recognize the same language, So it is possible to convert a multitape Turing machine to a single tape Turing Machine.

B. Non Deterministic Turing Machines

This machine can be achieved by converting the Deterministic part of Turing machines into Nondeterministic. Each time a nondeterministic move is made, it can be thought as a branch or "fork" to two simultaneously running machines. Each machine gets a copy of the entire tape. If any one of these machines ends up in an accepting state, then the input is said to be accepted. Nondeterminism does not affect the power of the Turing Machine model. Every Nondeterministic Turing Machine has an equivalent deterministic Turing Machine.

C. Restricted Turing Machines

This Turing Machine restricts the infinite length of the tape to a tape that is infinite only to the right. This machine is also forbidden to print a blank as the replacement tape symbol. With these restrictions, it is assumed that the Instantaneous Description consists of only non blank symbols and they always begin at the left end of the input. There are several configurations in which Turing machines can be restricted.

1) *Turing Machines with Semi Infinite Tapes:* In this kind of restricted Turing machine, there are no cells to the left of the initial head position.

2) *Multistack Machines:* These are based on generalizations of Pushdown Automata. Here the finite control can push and pop symbols from multiple number of stacks.

3) *Counter Machines:* A counter machine is sometimes considered a restricted multistack machine. It generally has same structure as multistack machine, but in place of stack is a counter. Counters hold nonnegative integers. Counter is incremented and decremented in the same way as push and pop operations are performed on the stack [12].

VI. Turing Machine and Artificial Intelligence

A Turing machine can be seen as a metaphor of the human brain. "Alan Turing's classical paper introduces the Turing Machine as a metaphor of a man in the process of computing a real number and whose human memory is necessarily limited" [7]. John Von Neumann raised the question how the brain, which is analogue, parallel and error-prone, could perform multistep computations without being swept away by biological noise [8]. A Turing machine can be made to perform the multi step computations as well by considering its various variants.

Turing machines remain adequate models of the conscious brain, and raise important novel issues for neuroscience. Among neuroscientists, however, this metaphor quickly fell into disrepute because it neglected aspects of the architecture of the brain that do not resemble those of a classical Turing device [7]. First, with a hundred billion processors, the architecture of the brain supports massive parallel processing [9]. Second, individual neurons exhibit complex and gradual behaviour unlike the digital circuits of a Turing machine, and populations of cells can operate with entire probability distributions [10]. Third, the brain is an evolved learning system whose architecture adapts at multiple timescales [11]. Despite these differences, the human brain can be slow and serial in executing some tasks. So concepts of implementation of Brain Turing Machines by the neural architectures are evolving and are an open research area.

VII. Conclusions

From this paper, it is concluded that Turing machine is a powerful model which can simulate the behavior of brain and can be an important part of neuroscience. Its power can be accounted by help of its structure which can be implemented in various ways. These are called its variants. Various variants of Turing Machines exhibit different functionalities to increase the computing capabilities but the power remains the same. Turing Machines are employed theoretically in various areas of research in complexity and computability theory. So, Turing machine is said to be a powerful automata which can compute complex problems and it is also said that if anything cannot be computed by a Turing Machine, it cannot be computed by a Computer as well.

References

[1] J. V. Neumann, "The General and Logical Theory of Automata", Design of Computers,Theory of Automata and Numerical Analysis, Volume 5, Sept. 20, 1948, Pasadena, California.

[2] A. Hodges, "Alan Turing: The Enigma", New York: Simon Schuster, 1983.

[3] H. L Lewis, "Turing Machines", Elements of Theory of Computation, Second Edition, Prentice Hall, Upper Saddle River, New Jersey, 1998.

[4] J. Teutsch, "A Universal Turing Machine", Available: http://people.cs.uchicago.edu/~odonnell/Teacher/Courses/UChicago/CMSC31100/UTM.pdf.

[5] M. L. Minsky: Computation: finite and infinite machines. Prentice-Hall, 1967.

[6] "Turing Machines" Available: http://www.math.ku.dk/~wester/turing.html

[7] A. Zylberberg, S. Dehaene, P. R. Roelfsema and M. Sigman, "The human Turing machine: a neural framework for mental programs", Trends in Cognitive Sciences, July 2011, Vol. 15, No. 7.

[8] V. Neumann, J. (1958) The Computer and the Brain, Yale University Press.

[9] D. J Felleman and V. Essen, D.C. (1991) Distributed hierarchical processing in the primate cerebral cortex. Cereb. Cortex 1, 1–47.

[10] A. Pouget, et al. (2003) Inference and computation with population codes. Annu. Rev. Neurosci. 26, 381–410

[11] D. V. Buonomano and M.M Merzenich (1998) Cortical plasticity: From synapses to maps. Annu. Rev. Neurosci. 21, 149–186

[12] Hopcraft, Motwani and Ullman, "Introduction to Turing Machines", Introduction to Automata Theory, Languages and Computation, 2nd Edition, 345-361, Addison Wesley, 2001.