IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 1, Ver. II (Jan. 2014), PP 26-30
www.iosrjournals.org

Model of Computation-Turing Machine

Pooja Tharani
(dept of CSE,Manipal University Jaipur,India)

Abstract : *In theoretical computer science and mathematics , the* ***theory of computation*** *is the branch that deals with how efficiently problems can be solved on a model of computation, using an algorithm. The field is divided into three major branches: automata theory, computability theory, and computational complexity theory.*

In order to perform a rigorous study of computation, computer scientists work with a mathematical abstraction of computers called a model of computation. There are several models in use such as ***Lambda calculus****,* ***Combinatory logic****,* ***mu-recursive functions*** *, but the most commonly examined is the Turing machine. Computer scientists study the Turing machine because it is simple to formulate, can be analyzed and used to prove results, and because it represents what many consider the most powerful possible "reasonable" model of computation . It might seem that the potentially infinite memory capacity is an unrealizable attribute, but any decidable problem solved by a Turing machine will always require only a finite amount of memory. So in principle, any problem that can be solved (decided) by a Turing machine can be solved by a computer that has a bounded amount of memory.*

I. Introduction

Let us look at some basic definitions –

1.1 Algorithms-a algorithm is defined as A finite sequence of simple instructions that is guaranteed to halin a finite amount of time. This is a very abstract definition, since We didn't specify the nature of this simple instructions.

For example an instruction can be "increment a number by one" or "Calculate the triple integral" We didn't specify the entity which can execute these instructions.For example is this entity a person, a computer, …

If it is a computer what is the processor type? How much memory does it have? …. ?

1.2 Abstract Machine- To make a more solid definition of algorithm we need to define an abstract (general) machine which can perform any algorithm that can be executed by any computer. Then, We need to show that indeed this machine can run any algorithm that can be executed by any other computer. Then, We can associate the notion of algorithm with this abstract machine or We can study this machine to find the limitations of computations.

Turing Machine- A conceptual model for general purpose computers proposed by Alan Turing in 1936 . A Turing machine has an unlimited and unrestricted amount of memory.A Turing machine can do everything a real computer can do.Nevertheless there are problems that a Turing machine cannot solve.

II. Turing machine specification

Components of Turing Machine:

[1].An unlimited length tape of discrete cells.

[2].A head which reads and writes on tape.

[3].A control device with a finite number of states which can instruct the head to read the symbol on the tape currently under head.and can instruct the head to write a symbol on the cell of the tape currently under tape.

[4]Move the head one cell to left or right.

[5]Change its current state.

III. Turing machine and Turing machine instructions

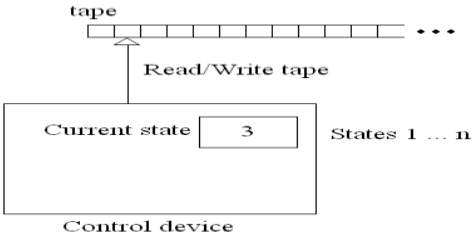

Fig3.1 a turing machine

Instructions of Turing Machine have the following format:

(Current State, Current Symbol, Write, Move L/R or No move, New State)

Ex: (2, 0, 1, L, 3) ,(3, 1, blank, N, 4), (1, #, 0, R, 7)

3.1The interpretation of the TM (Turing Machine) instructions:

Example3.1.1- (2, 0, 1, L, 3) Turing machine (the control unit of TM) is at state 2 and the current tape symbol is 0, write symbol 1 at current tape cell and go to state 3.

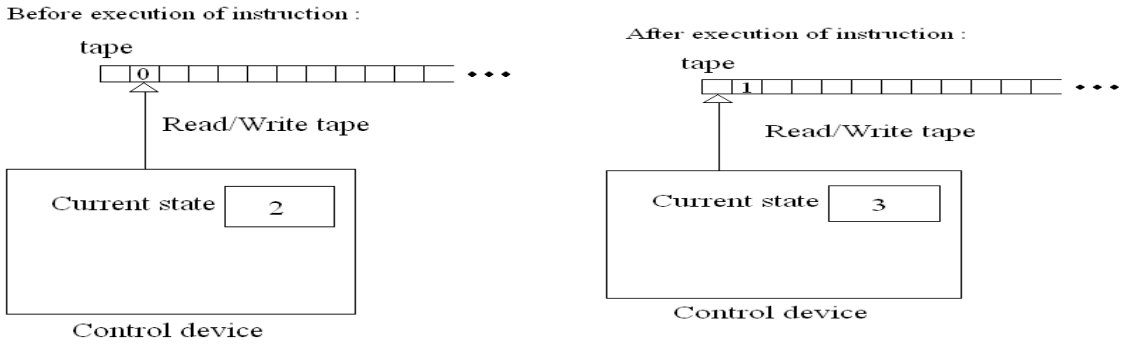

Fig3.2 Visualization of TM instruction (2, 0, 1, L, 3)

3.2- TM Conventions- We always use state 1 as the initial state. (That is the execution of the algorithm or program begins with stating of the TM being 1.The tape is used for recording input and output, one symbol per cell. Initially, the string to serve as input to our computation is recorded beginning from the leftmost tape cell.The output of a TM program or algorithm is the sequence of symbols on the tape when the TM halts on that program.

IV. Some basic examples on computation

Example4.1 Conversion from unary to binary . We consider the 4 state Turing machine illustrated below. The current state and input symbol are highlighted in yellow. We trace its execution.

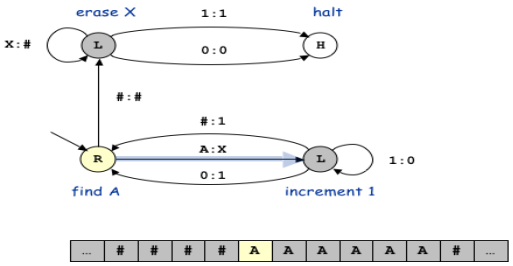

Figure 4.1.1

Since the input symbol is A, the Turing machine follows the appropriate transition arrow leaving the current state - the one labeled A : X. The Turing machine overwrites the input symbol with an X, changes state to the bottom right state, and moves the tape head one position to the left (since the new state is labeled with L). The illustration below shows the Turing machine at the end of this first step.

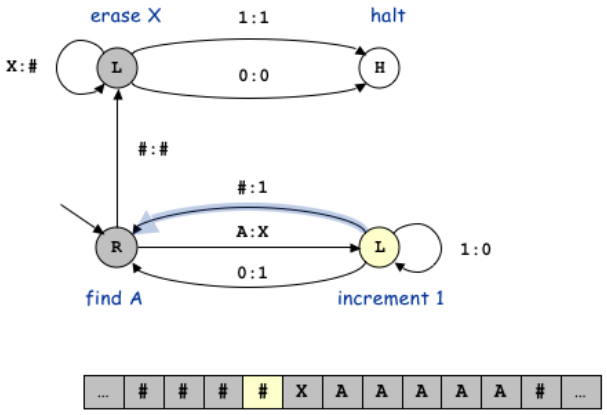

Figure 4.1.2

Since the input symbol is now #, the Turing machine follows the appropriate transition arrow leaving the current state -- the one labeled # : 1. This overwrites the current cell with a 1, changes the state back to the bottom left state, and moves the tape head one position to the right (since the new state is labeled with R).

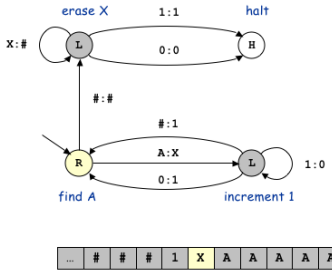

Figure 4.1.3

Since the input symbol is now #, the Turing machine follows the appropriate transition arrow leaving the current state -- the one labeled # : 1. This overwrites the current cell with a 1, changes the state back to the bottom left state, and moves the tape head one position to the right (since the new state is labeled with R).

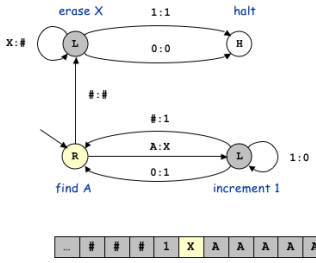

Figure 4.1.4

Here are the contents of the tape after the next several steps.

Table 4.1.5

…	#	#	#	1	X	A	A	A	A	A	#	…
…	#	#	#	1	X	X	A	A	A	A	#	…
…	#	#	#	1	X	X	A	A	A	A	#	…
…	#	#	1	0	X	X	A	A	A	A	#	…
…	#	#	1	0	X	X	A	A	A	A	#	…
…	#	#	1	0	X	X	A	A	A	A	#	…
…	#	#	1	0	X	X	A	A	A	A	#	…
…	#	#	1	0	X	X	A	A	A	A	#	…
…	#	#	1	0	X	X	X	A	A	A	#	…
…	#	#	1	0	X	X	X	A	A	A	#	…
…	#	#	1	0	X	X	X	A	A	A	#	…
…	#	#	1	1	X	X	X	A	A	A	#	…

Once all of the As are overwritten with Xs, the Turing machine erases all of the Xs (overwriting them with #s)

Table 4.1.6

…	#	1	1	0	X	X	X	X	X	X	#	…
…	#	1	1	0	X	X	X	X	X	X	#	…
…	#	1	1	0	X	X	X	X	X	X	#	…
…	#	1	1	0	X	X	X	X	X	X	#	…
…	#	1	1	0	X	X	X	X	X	X	#	…
…	#	1	1	0	X	X	X	X	X	X	#	…
…	#	1	1	0	X	X	X	X	X	X	#	…
…	#	1	1	0	X	X	X	X	X	X	#	…
…	#	1	1	0	X	X	X	X	X	#	#	…
…	#	1	1	0	X	X	X	X	#	#	#	…
…	#	1	1	0	X	X	X	#	#	#	#	…
…	#	1	1	0	X	X	#	#	#	#	#	…
…	#	1	1	0	X	#	#	#	#	#	#	…
…	#	1	1	0	#	#	#	#	#	#	#	…

Working-The Turing machine described above converts from unary to binary. That is, if the input consists of n consecutive A's, then the Turing machine prints the number n in binary to the left of sequence of A's (and overwrites the A's with X's). In the example above, the input consists of 6 A's and the Turing machine writes the binary number 110 to the tape.

To describe how this is accomplished, we first review an algorithm for incrementing a binary integer by 1: scan the bits from right to left, changing 1's to 0's until you see a 0. Then change the 0 to a 1.

The Turing machine repeatedly knocks off one A at a time and increments the binary number. Our Turing machine mimics this strategy. The initial state seeks out the next A, overwrites it with an X, and then transitions to the Increment state. The Increment state increments the binary integer by one (leaving the X's alone, changing 1's to 0's, until seeing a 0 or #, which it changes to a 1), and then transitions back to the Initial state. When all of

the A's are overwritten with X's, the Cleanup state replaces all of the X's with #'s, and the transitions to the Halt state.

Example 4.2 { (1, 1, 1, R, 2), (2, 1, 1, R, 2), (2, blank, blank, R, 3), (3, 1, blank, L, 4),
(4, blank, 1, R, 2) }
This program outputs the sum of two integers m and n given as input.The numbers are in base 1 (unary notation).
Examples of integers in unary notation:
1 = 1 2 = 11 3 = 111 4 = 1111 …. number n = n number of 1s.
The input on tape (the initial configuration):
1 1 b 1 1 1 1 b b b … state = 1 Inputs : operands 2 and 4.
The output on tape (when the program halts):
1 1 1 1 1 1 b b b … state = 3 output : 6 b stands for blank.
Exexcution of above example include instruction steps given below-
[1] . 1 1 b 1 1 1 1 b b … state = 1 Instruction1 which is going to be executed:(1, 1, 1, R, 2)
[2].1 1 b 1 1 1 1 b b … state = 2 Instruction2 which is going to be executed: (2, 1, 1, R, 2)
[3].1 1 b 1 1 1 1 b b … state = 2 Instruction 3which is going to be executed:(2, blank, blank, R, 3)
[4].1 1 b 1 1 1 1 b b … state = 3 Instruction4 which is going to be executed:(3, 1, blank, L, 4)
[5].1 1 b b 1 1 1 b b … state = 4 Instruction5 which is going to be executed:(4, blank, 1, R, 2)
[6].1 1 1 b 1 1 1 b b … state = 2 Instruction6 which is going to be executed:(2, blank, blank, R, 3)
[7].1 1 1 b 1 1 1 b b … state = 3 Instruction7 which is going to be executed:(3, 1, blank, L, 4)
[8].1 1 1 b b 1 1 b b … state = 4 Instruction8 which is going to be executed:(4, blank, 1, R, 2)
[9] 1 1 1 1 b 1 1 b b…state = 2 Instruction9 which is going to be executed:(2, blank, blank, R, 3)
[10] 1 1 1 1 b 1 1 b b … state = 3 Instruction 10 which is going to be executed:(3, 1, blank, L, 4)
[11]1 1 1 1 b b 1 b b … state = 4 Instruction 11 which is going to be executed:(4, blank, 1, R, 2)
[12].1 1 1 1 1 b 1 b b … state = 2 Instruction 12 which is going to be executed:(2, blank, blank, R, 3)
[13].1 1 1 1 1 b 1 b b … state = 3 Instruction13 which is going to be executed:(3, 1, blank, L, 4)
[14].1 1 1 1 1 b b b b … state = 4 Instruction 14 which is going to be executed:(4, blank, 1, R, 2)
[15].1 1 1 1 1 1 b b b … state = 2 Instruction 15 which is going to be executed:(2, blank, blank, R, 3)
[16].1 1 1 b 1 1 1 b b … state = 3 Instruction 16 which is going to be executed:(3, 1, blank, L, 4)
[17]. 1 1 1 b b 1 1 b b … state = 4 Instruction17 which is going to be executed:(4, blank, 1, R, 2)
[18].1 1 1 1 b 1 1 b b…state =2 Instruction18 which is going to be executed:(2, blank, blank, R, 3)
[19].1 1 1 1 b 1 1 b b … state = 3 Instruction19 which is going to be executed:(3, 1, blank, L, 4)
[20].1 1 1 1 b b 1 b b … state = 4 Instruction 20 which is going to be executed:(4, blank, 1, R, 2)
[21].1 1 1 1 1 b 1 b b … state = 2 Instruction 21 which is going to be executed:(2, blank, blank, R, 3)
[22]. 1 1 1 1 1 b 1 b b … state = 3 Instruction22 which is going to be executed:(3, 1, blank, L, 4)
[23].1 1 1 1 1 b b b b … state = 4Instruction which is going to be executed:(4, blank, 1, R, 2)
[24]1 1 1 1 1 1 b b b … state = 2 Instruction which is going to be executed:(2, blank, blank, R, 3)
[25].1 1 1 1 1 1 b b b … state = 3 There is no instruction starting with: (3 ,blank , ….) => HALT
Output : 1 1 1 1 1 1 b b b …

Example4.3 {(1, 0, 0, R, 2), (1, 1, 1, R, 2), (2, 0, 0, R, 2), (2, 1, 1, R, 2), (2, blank, 0, R, 3), (3, blank, 0, R, 4)}
Number of states: 4 Used alphabet : 0, 1
1 1 0 0 1 0 1 1 b b b … state = 1 instructions which is going to be executed will be (1, 0, 0, R, 2) , (1, 1, 1, R, 2),
(2, 0, 0, R, 2), (2, 1, 1, R, 2), (2, blank, 0, R, 3), (3, blank, 0, R, 4)
HALT. Output: 1 1 0 0 1 0 1 1 0 0 b … state = 4
INPUT: 11001011
What is the function that is being computed by this program?
OUTPUT: 1100101100
INPUT: 11001011
Input is base-2 presentation of number 203 and output is the base-2 presentation of number 812.
Thus,

$$f(x) = 4x$$

V. Type of problems input to the turing machine

Type of problems input to the turing machine can be categorized into
5.1 Decidable problems-Problems, for which we can't find an algorithm that answer all possible instances of the

problem. That is there is no TM program which answer all possible instances of the problem in a finite amount of time. For a decidable problem there is a program such that if an instance of the problem has solution, the program eventually halts with answer. But if there is no solution for that instance, the program will not ever halt. Can we consider such programs as algorithms?Answer: No, because they might not halt.

5.2 Un-decidable Problem-The problem of finding an integral solution for a collection of multi-variable polynomial equations, is not decidable.For example consider the following two instances of problem:

1-

$$x^2 + y = z$$
$$3y^3 + x^3 = 2z + 1$$

2-

$$xy + 3xz^2 + 62xyz = 14$$
$$13xy^5 + y - xz^3 = 9$$

Assume, we have a program which assigns all possible combination of 3 integers to variables x, y and z. For the first case there is at least one solution (x = 2, y = 1, z =5). Thus, the program will eventually stops. But for the second case we don't know if this system has a solution. If there is no solution for the second system, then the program never stops.

VI. Conclusion-

From a theoretical standpoint, we are primarily concerned with machines that perform finite computations and then halt. However, many practical applications involve programs that are designed never to terminate (operating system, air traffic control system, nuclear reactor control system) or produce an infinite amount of output (web browser, program that computes the digits of $\pi = 3.1415...$). The Turing machine model of computation extends to handle such non-terminating situations as well.Turing machines connect physics and mathematics (Turing's original motivation, thermodynamics of computation).

References

[1]. vars Peterson, 1988, The Mathematical Tourist: Snapshots of Modern Mathematics, W. H. Freeman and Company, New York.
[2]. Martin Davis editor, 1965, The Undecidable: Basic Papers on Undecidable Propositons, Unsolvable Problems and Computable Functions, Raven Press, New York, no ISBN, no card catalog number.
[3]. Alan Turing, 1937, On Computable Numbers, with an Application to the Entscheidungsproblem, pp. 116ff, with brief commentary by Davis on page 115.
[4]. Alan Turing, 1937, On Computable Numbers, with an Application to the Entscheidungsproblem. A Correction,