ISSN: 2456-8848, Vol. I, Issue. II, Sept-2017

International Journal
of
Linguistics & Computing Research

Introduction of Turing Machine

Shahbaaz khan
Department of Computer Science And Application
Dr. Harisingh Gour University, Sagar, India
shahbaazkhangolu@yahoo.com

ABSTARCT: Finite Automaton is computing device which work as an accepter for a regular language. Similarly, context free language is accepted by the machine called Push Down Automat (PDA). Turing Machine is more powerful then these machine as it is capable to accept language generated by type 0 grammar. This paper introduces Turing machine and its various variants.

Keywords: Tuning Machine,

I. INTRODUCTION

Alan Mathison Turing (1912-1952), was very influential in the development of computer science and providing a formalization of the concept of the algorithm and computation with his famous Turing machine, which played a significant role in the creation of the modern computer. Turing first described the Turing machine in his 1936 article "On Computable Numbers, with an Application to the Entscheidungs problem" [1].

Finite Automaton (FA) is an accepter, regular grammar generates language and finite automata accept it. In the same way context free grammar (CFG) generates language and PDA accepts it. But TM is an accepter, generator and transducer; it can accept any language (type 0 grammar).

Anything can be computed by a TM, limitations that hold on Tm also provably apply to our real world machines. These models gave rise to a lot of theoretical analyses that resulted in a profound understanding of aspects of 'computability'.

A Turing machine (TM) is an accepting device which accepts the languages (recursively enumerable set) generated by type 0 grammars. It was invented in 1936 by Alan Turing.

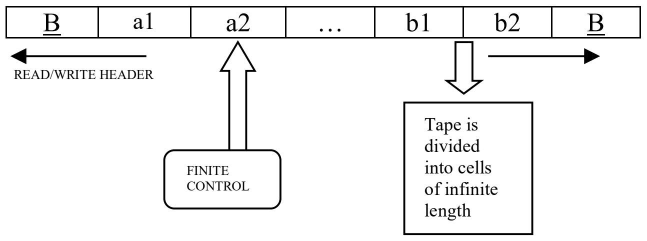


Figure 1: An pictorial representation of Tape of TM

The Turing machine can be thought o as finite control connected to a R/W (read/write) head. It has one tape which is divided into a number of cells, as shown in figure 1.

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

1. A new symbol to be written on the tape in the cell under the R/w head.
2. A motion of the R/W head along the tape, either the head moves one cell left (L) or one cell right (R).

3. The next state of the automation.

3. Whether to halt or not.

II. MATHEMATICAL DEFINITION

A TM can be formally described as a 7-tuple (Q, X, ∑, δ, q0, B, F)

Q is a finite set of states

X or ┌(Tao) is the tape alphabet. Toe is a set of symbols which are written on tape.

∑ is the input alphabet

δ is a transition function; δ : Q × X → Q × X × {Left_shift, Right_shift}.

q0 is the initial state

B is the blank symbol

F is the set of final states

The working of Turing Machine is shown in figure 2.

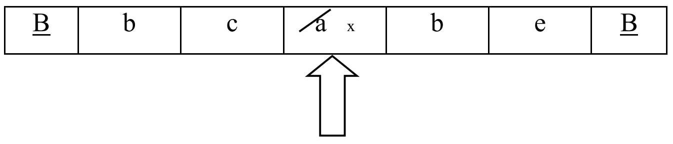


Figure 2 : Working of TM

δ : Q × X → Q × X × {Left_shift, Right_shift}.

On a state Q taking input symbol X (Tao) or (┌). After reading writing is mandatory. If we do not want to make any changes then we write can write the same symbol. Move is mandatory.

(L/R) -> Direction of move left or right.

Example: consider the language L = {$a^n b^n$: n>=1} then Turing Machine can be designed as below:

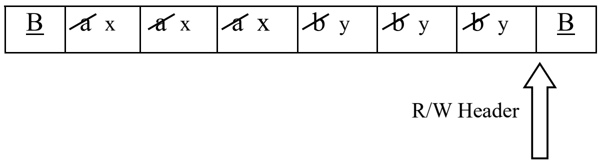


Figure 3 : Pictorial representation of tape

Header has the capability to move left or right and read and writes. So that we can remember how many 'a' has been processed and how many 'b' has been processed.

'a' is translated to 'x' and we will move forward until we find its corresponding 'b' which is then translated to 'y'.

'a' is translated to 'x' and we will move forward until we find its corresponding 'b' which is then translated to 'y'. Transition is shown in figure 4.

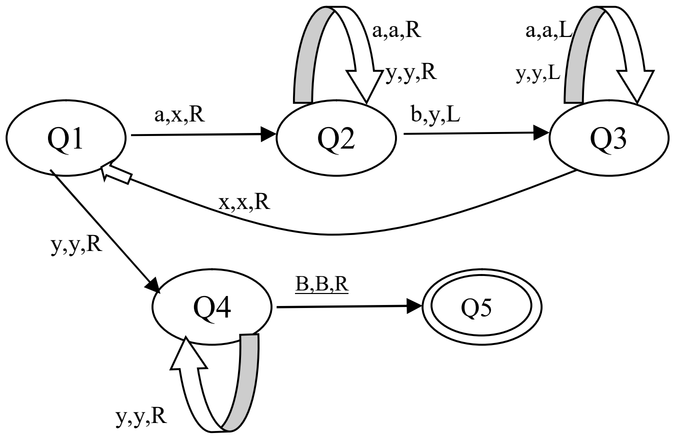


Figure 4: transition diagram of Tm accepting L={$a^n b^n$: n>=1}

- When 'a' are finished 'b' will also be finished.
- It accepts number of 'a' = number of 'b'.

III. APPLICATIONS OF TURING MACHINE

FA is an accepter in which regular grammar generates language and FA automata accepts it. CFG generates CFL and PDA accepts it.

Turing machine is an accepter, generator and transducer.

i. TURING MACHINE AS AN ACCEPTER : It can accept any language.

ii. TURING MACHINE AS GENERATOR: Like Mealy and Moore machine it has the capability to generate language/strings.

iii. TURING MACHINE AS TRANSDUCER :- It can solve any mathematical function, for example addition, subtraction, division, multiplication , log etc

If a problem is solved by a computer like solving a numerical problem or solving an algorithm then that problem can also be solved by the Turing machine.

IV. VARIATIONS OF TURING MACHINE

Turing machine was further explored and different variations are proposed. Some of them are following:

1. Multi-track Turing machine.
2. Multi-tape Turing machine.
3. Non-deterministic Turing machine

4. Linear bounded Turing machine
5. 2 Stack Turing machine , etc

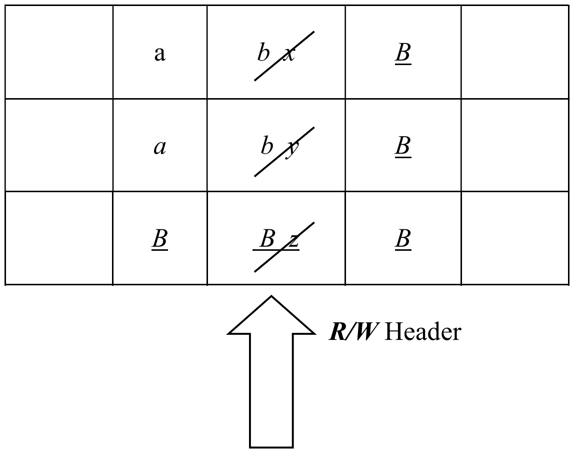
MULTI-TRACK TURING MACHINE

Multi-track Turing machines, a specific type of Multi-tape Turing machine, contain multiple tracks but just one tape head reads and writes on all tracks. A Multi-track Turing machine can be formally described as a 6-tuple (Q, X, ∑, δ, q0, F) . Shown in figure 5

- Q is a finite set of states
- X is the tape alphabet
- ∑ is the input alphabet
- δ is a relation on states and symbols where
- δ(Qi, [a1, a2, a3,....]) = (Qj, [b1, b2, b3,....], Left_shift or Right_shift)
- q0 is the initial state

F is the set of final states.

How multi-track Turing machine works.



Q(b,c,B)=(Qnext,x,y,z)

Figure 5: Pictorial representation Multi-track TM

MULTI-TAPE TURING MACHINE

Multi-tape Turing Machines have multiple tapes where each tape is accessed with a separate head. Each head can move independently of the other heads. Initially the input is on tape 1 and others are blank. As shown in figure 6

Q is a finite set of states

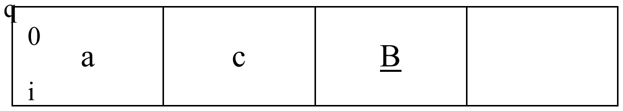
X is the tape alphabet

B is the blank symbol

δ is a relation on states and symbols where

δ: Q × Xk → Q × (X × {Left_shift, Right_shift, No_shift })^k

Where there is k number of tapes



q0 is the initial state

F is the set of final states.

Pictorial representation of a Multi-tape Turing machine is shown in figure 6

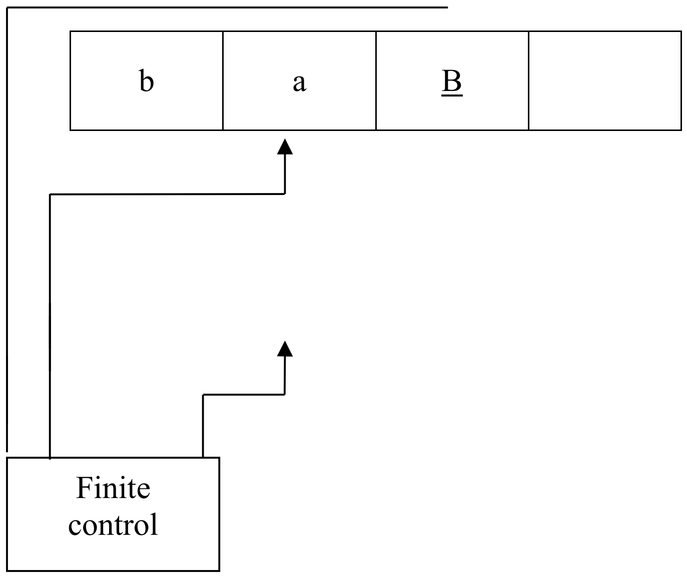


Figure 6: Pictorial representation of Multi-tape TM

NON-DETERMINISTIC TURING MACHINE

In a Non-Deterministic Turing Machine, for every state and symbol, there are a group of actions the TM can have. So, here the transitions are not deterministic.

A non-deterministic Turing machine can be formally defined as a 6-tuple (Q, X, ∑, δ, q0, B, F) where −

Q is a finite set of states

X is the tape alphabet

∑ is the input alphabet

δ is a transition function;

δ : Q × X → P(Q × X × {Left_shift, Right_shift}).

q0 is the initial state

B is the blank symbol

F is the set of final states

LINEAR BOUNDED TURING MACHINE

The computation is restricted to the constant bounded area. The input alphabet contains two special symbols which serve as left end markers and right end markers which mean the transitions neither move to the left of the left end marker nor to the right of the right end marker of the tape. As shown is figure 7

Q is a finite set of states

X is the tape alphabet

∑ is the input alphabet

q0 is the initial state

ML is the left end marker

MR is the right end marker where MR ≠ ML

δ is a transition function which maps each pair (state, tape symbol) to (state, tape symbol, Constant ‘c’) where c can be 0 or +1 or –

F is the set of final states.

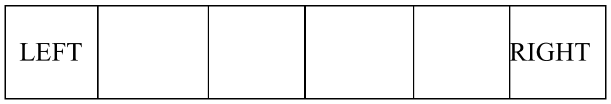


Figure 7: Pictorial representation of Linear Bounded TM

V. IMPORTANCE OF TURING MACHINE

The Turing machine can compute anything that can be computed. It is the very definition of computation and the fundamental tool for reasoning about computers. Modern computers are example of such a device and all modern programming languages are Turing complete (they can simulate any algorithm that could be run on a Turing machine).

If we want to know the exact functionality of the computer and also we know whether a particular problem is solvable or not by our computer so for what we convert that physical machine to a mathematical model TM.

Turing formulated the Turing machine in order to capture everything hat computers can do (without using Quantum Mechanics, by his own admission).

The “everything” statement is known today as the Church-Turing thesis and is widely believed to hold.

EXAMPLE: - If you take some physical system that performs computation, you can simulate it numerically (with approximation) on a Turing machine.

Turing machine is an abstract mathematical idea – not a literal wires and metal machine.

It can always be configured to give a particular output for a particular input. But this simple idealized device requires infinite memory so it’s not possible to build a physical Turing machine.

However, a physical device is said to be Turing machine it can simulate any algorithm that could be run by the idealized Turing machine.

There are many variations of Turing machine like multi-header Turing machine, multi-track Turing machine etc. It is possible that variations of Turing machine can work on different speed (Ex- 2 header can work faster).

But we cannot change the power of TM. *If a language is accepted by a variant of TM then it can be accepted by any other variation of Turing machine.*

VI. SEVERE CHANGES FROM FA TO TM

- The header has the capability to read and write.
- The tape is a two way infinite tape divided into cells.
- Each cell contains one symbol.
- ┌ (Tao) is a set of all symbols written on the tape.
- B [Blank symbol] means empty cells.
- Transition function, δ : Q × X → Q × X × {Left_shift, Right_shift}.

o On a state Q taking input symbol X or ┌ . After reading writing is mandatory. If we do not want to write new symbols or to skip, then we can write the same symbol.

o (L/R) means direction of move. The header can move one cell to the left or one cell to the right.

PDA plus one stack becomes a Turing machine, since PDA has one stack but if we add one more stack, it becomes a Turing machine. Finite automaton (FA) plus two stacks also becomes a Turing machine.

CONCLUSION

Turing Machine is more powerful then finite automaton and push down automat as it is capable to accept language generated by type 0 grammar. This paper introduces Turing machine and its various variants. Turing machine can also work as generator and transducer

REFERENCE

[1]. Turing, A.M. (1936), "On Computable Numbers, with an Application to the Entscheidungs problem", Proceedings of the London Mathematical Society, 2 (published 1937), 42 (1), pp. 230–65.

[2]. Aho, A. V., Lam, M. S., R. Sethi, R., et al. 2007. *Compilers: Principles, Techniques, and Tools*. 2nd ed., Addison-Wesley, New York.

[3]. Hopcroft, J. E., R. Motwani, R., and Ullman, J. D. 2007. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, New York.

[4]. Linz, P., 2010. *An Introduction to Formal Languages and Automata*. 4th ed., Narosa Publishing House, New Delhi.

[5]. Goddard, W. 2010. *Introducing the Theory of Computation*. First India Edition, Jones and Bartlett India Pvt. Ltd. 1)

[6]. Mishra K.L.P, CHANDRASEKARAN N.: Theory of Computer Science: Automata, languages and Computation.