Automata Theory Write-Up

Nelson Mwangi-ICS A -134979

July 2022

1 Turing Machines

1.1 Introduction

Turing Machines are devices that accept languages generated by type 0 grammars (Recursively enumerable Sets) that were invented in 1936 by Allan Turing. For there are problems that cannot be solved using a computer that are called "undecidable", the venerable formalism for computers the Turing Machines are brought into play. However, Turing machines too cannot solve all problems. Turing Machines are made up of a tape of infinite length that is further divided into an infinite number of cells on which read and write operations can be performed. Thus, the Turing machine is like a finite automaton but with infinite memory. It is a more accurate model of a general-purpose computer.

The cells either contain an input symbol or a special symbol called a blank. A multi-directional head pointer is used to define the cell being currently read. A state register is used to store the state of the Turing Machine. Initially, the tape only contains the input string and is blank everywhere else.

After an input symbol is read, it is replaced by another symbol thus its internal state is also changed. The head pointer then moves either right or left until the final state is reached where the string is accepted or until the string is rejected. If neither the accepting state nor the reject state is achieved, the machine will go on forever without cease.

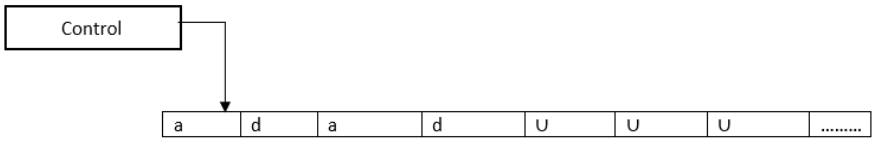

Figure 1: Description of a Turing Machine

1.2 Formal Definition of a Turing Machine

The Turing machine is formally expressed as a 7-tuple (Q, X, , , q0, B, F) with Q, X, are all finite sets with:-

- "Q" as the set of states
- "" as the set of input alphabet without the blank symbol
- "X" is the tape alphabet of symbols which can be written on the tape
- "" is the transition function that serves as the heart of the Turing Machine since it dictates the movement of the machine from one step to the next.
- "q0" is the start state of the Turing machine
- "B" is the blank symbol the fills all the cells initially since they are all blank at the start
- "F" is the set of Final states where a string is accepted if the sequence is accepted.

1.3 Differences between Turing Machines and Finite State Machines

The following are the differences between the Turing machine and the Finite Automata:

- The Turing machine can both read and write from the tape while Finite Automata can only read
- The tape in a Turing Machine is infinite thus it as infinite memory while the Finite Automaton has limited memory.
- In a Turing machine the accept state and the rejecting state take effect immediately while the Finite Automaton does not.

1.4 Example of a Turing Machine

Design a Turing Machine that accepts a string of the following order

$$L = 0^N 1^N$$

The string above is of a language that is only accepted if the number of '1's and '0's is the same. To design the Turing Machine for the above string, we can develop an algorithm to understand the flow of computing.

In this case, we will assume

$$'N's = 3$$

to generate the string: - In the tape, we will substitute the input alphabet '0'

	0	0	1	1	u	..	

Table 1: Example String

with 'X' as the tape alphabet and '1' with 'Y' as its tape alphabet. Consequently, we can generate the transition function '' for the Turing Machine

$$(q0, 0) = (q1, X, R)$$

The above function states that if the read symbol in q0 is '0', we replace it with 'X' before moving the head right to q1.

$$(q1, 0) = (q2, 0, R)$$

The above transition function also checks whether the input alphabet is '0' but does not change the '0' to the tape alphabet.

$$(q2, 1) = (q1, Y, L)$$

The above function checks the input alphabet if it is equal to '1' and substitutes it with 'Y' as the tape alphabet. The process is repeated until completion therefore the pointer head will move to leave most input alphabet that is a '0'

$$(q1, 0) = (q3, X, R)$$

In this case another '0' is found and switched with a tape alphabet 'X' and the process repeats again.

$$(q3, 1) = (q0, Y, L)$$

One last cycle will be done to ensure all input alphabet has been switched.

$$(q3,) = (q4, , R)$$

The above function leads to the halt state that is the accept state in a Turing Machine.

With the above information, it is possible to formally describe the Turing Machine in the following order:

$$Q = q_0, q_1, q_2, q_3, q_4$$

$$X = 0, 1, X, Y, \emptyset$$

$$\Sigma = 0, 1$$

$$q_0 = q_0$$

$$B = \emptyset$$

$$F = q4$$

1.5 Time and Space Complexity of a Turing Machine

Given a Turing machine, the space complexity can be based on the number of tape cells written- that is the amount of space, and the time complexity which is the count of tape moves when the machine is started for a set of input symbols. The size of input is the most suitable parameter to be considered when solving the time and space complexity of a Turing Machine. Time complexity all reasonable functions

$$T(n) = O(nlogn)$$

Turing Machine's space complexity

$$S(n) = O(n)$$

1.5.1 Time Complexity

In a Turing Machine, the time complexity is the number of times the tape moves when the machine is filled with some input. To elaborate this statement, consider the following explanation. Turing machine T has two-way infinite tapes with one of these tapes containing the input and all other tapes including this tape can be performed a read or write operation on. Time Complexity is therefore defined as:-

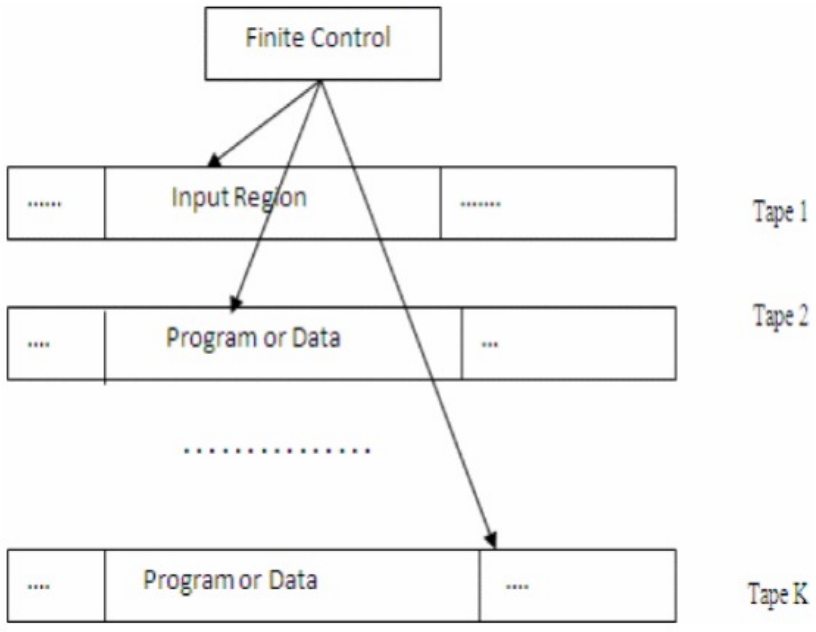

Figure 2: Turing Machine T

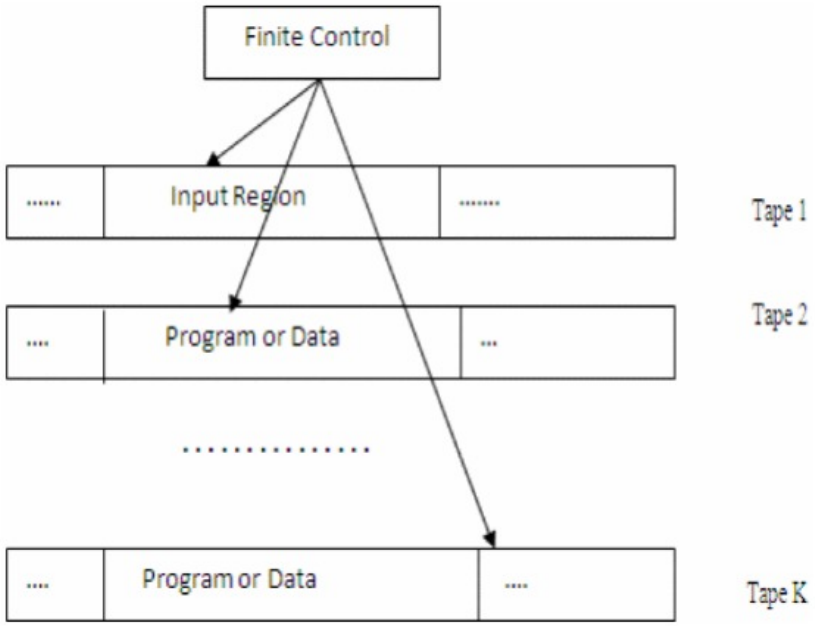

Figure 3: Turing Machine M

1.5.2 Space Complexity

Considering an off-line Turing device shown below ,shows M of K tapes. M has k-l One-way infinite tapes as well as a read-only tape. Complexity of space will be defined as:

If for every input string of length n, Turing machine M scans at most S(n) cells on the chosen write tape then it is said to be S(n) space bound Turing machine thus the language recognised by such a machine is of the space complexity S(n)

1.6 Turing Machine Halting Problem

A Turing machine and the input string w are the inputs.

Does the Turing machine complete the string W's computation in a limited number of steps? Either yes or no must be given as a response.

Proof We will first suppose that a Turing machine exists to address this issue, and then we will demonstrate how it is internally inconsistent. This Turing computer, which outputs a "yes" or "no" in a set length of time, will be referred to as a halting machine. The result is "yes" if the halting machine completes in a limited period of time, and "no" otherwise. The block diagram of

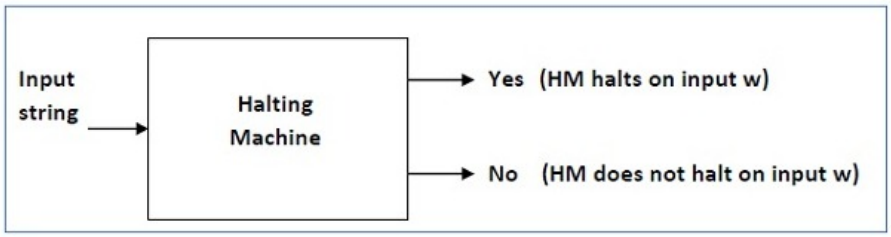

Figure 4: Caption

a halting device is shown below. We will now create an inverted halting machine (HM) in the form of

Continually loop if H returns YES. If H gives a NO response, stop. On the other hand the inverse's results give a contradicting solution.

1.6.1 Proof by contradiction

Can we create a computer that, given a program, can determine whether or not that program will always cease on a specific input?

Solution: Assume that we are able to create the type of machine known as HM(P, I), where HM stands for machine/program, P for program, and I for input. The machine HM will inform the user if program P halts or not after receiving both parameters as input. If we are able to create such a program, we are then able to write another program, which we refer to as CM(X), where X is any program (given as an input), and we write it in accordance with the definition of CM(X) shown in the figure.

1.7 Rice Theorem

Any non-trivial semantic attribute of a language that is understood by a Turing computer is deemed undecidable according to the Rice theorem. All Turing machines that meet a particular property, P, have the same language. P is a collection of languages, which is a property of languages. It is said that L satisfies the property P if any language (L P) is a member of P.

If a property is met by all recursively enumerable languages or if it is not satisfied by any recursively enumerable languages, it is said to be trivial.

Some recursively enumerable languages satisfy a non-trivial characteristic, whereas others do not. Formally, both of the following qualities hold in a non-trivial property where L P:

- Property 1: Either (M1¿, M2¿ L) or (M1¿, M2¿ L) exist Turing machines that can recognize the same language.
- Property 2: There are two Turing machines, M1 and M2, but only M1 can detect languages, i.e., M1 ¿ L and M2 ¿ L.

1.8 Variations of the Turing Machines

1.8.1 Multiple-Tape Turing Machine

A multi-tape machine has multiple tapes that can be accessed by a different head pointer. Each of these pointer heads are independent thus move individually. Initially, the first tape receives the input while the rest remain blank. Once the first input is taken, it is consequently filled on the first tape while the rest remain empty until the machine reads consecutive symbols under its heads. It can be described as a 6 tuple

$$(Q, X, B, , q0, F)$$

. Of the listed set elements in the 6-tuple, the input language is eliminated compared to the original 7-tuple Turing machine

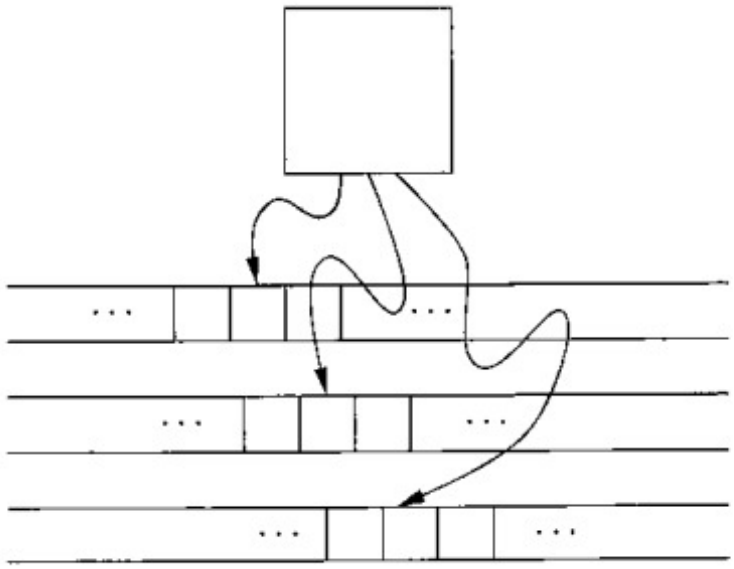

Figure 5: Multi-tape Machine

1.8.2 Two-way Infinite Tape Turing Machine

Two-way infinite tape of infinite tape Turing machines are unlimited in both left and right directions. a two-way, endless tape they are more-or-less one way infinite Turing machines. The pointer head in this case can move in four directions :- To the right, left , up or down. This type of Turing Machine can also b simulated by a single one-dimensional tape Turing machine.

1.8.3 Multi-Track Turing Machine

This is a deviant of the multi-tape Turing Machine variant. However, it is different in the sense that it has only one pointer head that does the read and write operations instead of the Multiple heads for each tape. Recursively enumerable languages are accepted, just like they are for single-track, single-tape Turing machines. As much as it has lesser heads, it has equal expressive power to the multi-tape Turing machine. It can also be described as a 6-tuple

$$(Q, X, , , q0, F)$$

like the original

1.8.4 Semi-Infinite Tape Turing Machine

As its name dictates, it is partially infinite since it has a definite left end however it does not have a fixed right end. The initial left-end is signified by a marker. It has two unique states known as the accept state and the reject state. The input is accepted by the TM if it ever changes to the accepted state; conversely, if it ever changes to the reject state, the input is rejected by the TM. Sometimes it runs forever without being approved or refused for certain specific input symbols

1.8.5 Non-Deterministic Turing Machine

In the non-deterministic Turing Machine, there are several possible TM actions for each condition and symbol. Thus, the transitions in this case are not deterministic. A tree of configurations leading from the initial configuration can be achieved by a non-deterministic Turing machine's computation. In the non-deterministic TM an input is accepted if there is at least one node that is accepted otherwise it is rejected. The non-deterministic Turing machine is referred to as a Decider if all branches of the computational tree come to an end on all inputs, and the input is also rejected if all branches of the computational tree reject some input.

1.9 Significance of Turing Machines in Modern Computer Science

In modern day computer science, the Turing machine, it's structural and surrounding theories have been a key pillar. In some cases, Turing machines have been identified as the father of modern computer technology. The potential of the Turing Machine was, and is still, evident from the beginning when Allan Turing made the invention. One significant result of Turing's initial investigations was that his Turing Machine could compute any computable number. Of the very many uses and significance the Turing Machine has here are some of the most outstanding milestones Turing Machines have led to:

1.9.1 Intelligence and Consciousness

Regarding our daily activities, most real-world problems fall under two general categories – important things and less important or even negligible. Consequently, we humans have established that we are intelligent beings since we are able to determine genuine purpose – an end goal that is contemplated on before being acted upon. As a result, it has become a theory whether this implementation of intelligence can be applied to computers. This theory, over time, sparked the idea of creating human-like sentience using computer technology fashioned in a common term Artificial Intelligence. This idea has become more of reality in recent days and has been decorated with modern day ideas and lifestyle to produce results that exceed our own expectations and human capability. AI is the fusion of the idea of being able to sort out priority depending on the desire to achieve an end goal together with the process of collecting, sorting, and storing large amounts of data. This has led to creation of, albeit not-yet-perfect, man-made intelligence – thus the name Artificial Intelligence .

1.9.2 Information Processing and Phenomenology

As a result, AI has been misinterpreted by many following heir inability to distinguish intelligence and sentience. Human-like sentience requires intelligence, but a machine is incapable of being sentient itself. This is because of modern science limitations. As such artificial intelligence do not have a mind of their own

2 Programming Languages

All languages are recognized by the Turing machine even though they may all be recursively enumerated. Recursive refers to constantly using the same set of rules, whereas enumerable refers to a list of items. The TM accepts computations for addition, multiplication, division, subtraction, and many more operations.

others.

If a Turing computer takes all input strings w and terminates on any of them, the language

is referred to as being Recursive or Decidable. Every decidable language is Turing acceptable.

If all yes instances to a decision issue P's language L, then the problem can be decided.

is a decision.

For each input string in a language that may be decided, the TM halts either at the

As seen in the accompanying graphic, the states are either accept or refuse. Example Check to see whether the following issue can be resolved. Is m a prime number? Solution Prime numbers are 2, 3, 5, 7, 11, and 13, among others. Divide each integer between "2" and "m" by the number "m." If any of these

numbers leave a remainder of 0, the "Rejected state" is reached; if not, the "Accepted state" is. Therefore, you might respond to this question with "Yes" or "No." It is a decidable problem as a result.

3 References

@articlewegner2003computation, title=Computation beyond Turing machines, author=Wegner, Peter and Goldin, Dina, journal=Communications of the ACM, volume=46, number=4, pages=100–102, year=2003, publisher=ACM New York, NY, USA @articlecore1998theory, title=Theory of Computation, author=Core, AITB03, year=1998