Strange Turing Machines

By: Eric Robles and Luke Skinker

Invention of Turing Machine

A century ago, the mathematician, David Hilbert, was searching for an answer to the question:

> Is there a systematic approach to prove mathematical theorems?
> Can any mathematical problem be solved with an algorithm?

Mathematicians executed algorithms themselves to precisely describe how to solve problems.

In 1936, Alan Turing invented a machine that was able to run these algorithms. It was so simple that was universally computational: it could execute any possible algorithm (Rachid Guerraoui)

Turing Machine Equivalents

The standard Turing machine has been modified many times to suit a particular need, these Turing machines that have been modified are equivalent Turing machines.

To be an “equivalent” Turing machine, in this case, means to be something that may be faster, or less resource intensive than a standard infinite single-tape Turing machine, but it can never be more powerful than the regular Turing machine.

The idea that a variant of a Turing machine will never be more powerful than a standard Turing machine is explained within the Church-Turing thesis, which states something is computable if and only if a standard Turing machine can compute it.

What We Will be Talking About

Since there are a large number of variants, the “strange” Turing machines that we will be presenting about today are…

- Universal Turing Machine
- Read-Only Turing Machine
- Multi-Track Turing Machine
- Oblivious Turing Machine

Universal Turing Machine

The Universal Turing Machine (UTM) was introduced by Alan Turing in 1936-37.

How does it relate to theoretical computer science?

Since its invention, it has been the most used model of computation in computability and complexity theory. It is a mathematical tool that is equivalent to a digital computer with a fixed program: computes a function from an input string over an assigned alphabet.

Any modern computer that is capable of copying a program file from one place to another, and later run this program, follows this architecture.

How is the UTM Applied to Real Life?

Turing Machines are considered as “hardwire”, which means that they only execute one program. Real computers are re-programmable.

The UTM is a re-programmable machine that simulates any other TM, *M*.

Input of UTM:

- Description of the transitions of *M*
- Input String of *M*

In other words, the description of *M* is a program, and the UTM simulates the program in a software.

How Does the UTM Work?

The UTM simulates another TM by taking it as an input. The UTM takes in the description of the <TM> and the <input of its tape> as its parameters. The input is presented in a binary form.

Draw picture and explain how the tape moves and is encoded

Draw example of UTM and talk about recognizable vs. undecidable

The Halting Problem

Anything that can be computed can be computed by an UTM. Although, there is a problem determining whether an arbitrary TM will halt on a particular input, or run in an infinite loop. In general, this tends to make the machine recognizable, but undecidable.

In 1936, Alan Turing prove that generating an algorithm that will solve the halting problem does not exist.

Draw diagram and explain the halting problem

In other words, it is impossible to build a TM that can solve the halting problem. This is a problem that is beyond the limits of computation.

Read-Only Turing Machine (ROTM)

- Similar to a regular Turing Machine, however a read-only Turing machine can only read tape but cannot write on it
- One could see that a read-only Turing machine is very similar to a DFA, due to the fact that it would never have to backtrack through the tape
- Another name for this Turing machine is actually Two-way Deterministic Finite Automaton
- From this variant of a standard Turing machine there are other variants, like the read-only right moving Turing machine and others

What we should also ask from something like this is, what would we use a Turing machine that can only read from tape for?

ROTM and What We Use It For

- The Universal Turing machine is able to create other Turing machines, but it must first make sure that the Turing machine being made is able to accept the definition of the Turing machine that is making it. This requires a Read-Only Turing machine to ensure that it will accept.
- Read-Only Turing machines are also important within quantum computing, due to the fact that quantum computing uses Universal Turing machines for some calculations.

ROTM: Question

If Read-Only Turing machines are very similar in functionality and computing power to a DFA, what kind of language will it be able to parse?

ROTM and Regular Language

Read-Only Turing machines can only parse regular languages, however this leads to an interesting conclusion about Read-Only Turing machines.

- We know that regular languages are recognized by finite automaton
- Read-Only Turing machines are extremely similar to DFAs and have the same computational power, so it would need some way to only deal with finite amounts of information

This has to mean that a Read-Only Turing machine will only be able to store a finite amount of information by having markers set up to indicate the end of the input or have specific instructions on how to deal with blank tape space so that it does not go on infinitely.

Multi-Track TM vs. Multitape TM

A Multi-Track TM is a type of Multi-tape TM that contains multiple tracks.

A Multi-Tape TM contains *n* number of tapes and *n* number of heads that move independently along *n* number of tracks.

In contrast, the Multi-Track TM contains multiple tracks, but only one head that reads and writes on all tracks simultaneously. The machine contains *n* symbols in the position of *n-track.*

It accepts recursively enumerable languages like normal single-track TMs.

Definition of a Multi-Track TM

A Multi-Track TM is defined as 6-tuple:

$M = < Q, \sum, \Gamma, \delta, q0, F >$

Q -> finite set of states

$\sum$ -> input alphabet

Γ -> tape alphabet, where $B \in \Gamma$ and $\Sigma \subseteq \Gamma$ B -> Blank space

δ -> $Q \times \Gamma \rightarrow Q \times \Gamma \times \{ L, R \}$

q0 -> Initial State

F -> Accepting State

Multi-Track TMs Are Equivalent to Single-Track TMs

Let *M* be a single-track TM -> $M = < Q, \sum, \Gamma, \delta, q_0, F >$

For every *M*, there is a multi-track TM, $M' \mid L(M) = L(M')$

$M' = < Q', \sum', \Gamma', \delta', q_0', F' >$ $\sum'$ x k times for k-track Γ' x k

$\delta' = (q_i,[a_1,a_2,\ldots a_k]) = '(q_i, <a_1,a_2,\ldots a_k>)$

Multi-track TMs are equivalent to single-track TMs, but just a different representation.

Oblivious Turing Machine

Oblivious Turing machines are different from the standard Turing machine because the head of the machine operates with regards to time, not the input.

This means that the head of the machine will work at a pace that is predetermined and will follow this predetermined pacing as it moves through the input.

This independence from the input, however, does make the machine take longer than the standard Turing machine.

Oblivious Turing Machine Facts

- The Oblivious Turing machine is set up so that even if there are multiple tapes to deal with, the heads will move at the exact same time and in the exact same motion. This is actually the trait that gives the name “oblivious” to this Turing machine.
- Oblivious Turing machines have seen use in cryptography, where someone would use an Oblivious Turing machine to actually make the something encrypted.
- One type of a cryptographic application for an OTM is something called Oblivious RAM, which is a type of software protection by preventing the gathering of information from how memory is distributed, made by Goldreich in 1987.

Standard TM vs. Oblivious TM

The best way to show the comparison between these two equivalent Turing machines is to show how fast they run on the same input. Pippenger and Fischer, authors of the famous “Relations Among Complexity Measures” article in 1979, were able to prove:

- Standard Turing machines, with n inputs, will take O(n) amount of time to complete
- Oblivious Turing machines, with n inputs, will take O(n log n) amount of time to complete

This time difference is obviously due to the independence that the Oblivious Turing machine has over the input while the standard Turing machine.

“Strange” Turing Machines

Turing machine is the simplest implementation of “computing” and informally we say that if something can be computed then it can be done using a Turing machine. It might take a little longer to work out an answer with a Turing machine than the latest PC but that’s not the point. If it can be computed we accept the premise that it can be computed by a Turing machine even if we have to wait a while for it to finish.

Work Cited

- Banitt, Tracy D. "The Halting Problem." *Macalester Journal of Philosophy* 4th ser. 5.1 (1994): 12-16. Macalester Journal of Philosophy. Web.
- Copeland, B. Jack. "The Church-Turing Thesis." *Stanford Encyclopedia of Philosophy*. Stanford, 2002. Web. 30 Nov. 2016.
- Costas, Busch. "A Universal Turing Machine." *University of Toledo*. University of Toledo, n.d. Web. 26 Nov. 2016.
- Guerraoui, Rachid. “The Universal Turing Machine (Ft. Rachid Guerraoui).” *YouTube.* YouTube, 17 Nov. 2016. Web. 25 Nov. 2016
- Jones, Neil D. *Computability and Complexity: From a Programming Perspective*. N.p.: n.p., n.d. *Google Books*. MIT Press, 1997. Web. 30 Nov. 2016.
- Kamvysselis, Manolis. "Universal Turing Machine." *Universal Turing Machine*. Massachusetts Institute of Technology, n.d. Web. 26 Nov. 2016.
- Lipton, R. J. "Oblivious Turing Machines and a “Crock”." *Gödel’s Lost Letter and P=NP*. N.p., 28 July 2009. Web. 30 Nov. 2016.

Work Cited

- "Multi-track Turing Machine." *Wikipedia*. Wikimedia Foundation, n.d. Web. 29 Nov. 2016.
- "Oblivious RAM." *Wikipedia*. N.p., 12 Feb. 2016. Web. 30 Nov. 2016.
- Pippenger, Nicholas, and Michael J. Fischer. "Relations Among Complexity Measures." *Journal of the Association for Computing Machiner* 26.2 (1979): n. pag. Web. 30 Nov. 2016.
- "Read-Only Turing Machines." *Wikipedia*. N.p., 12 Feb. 2016. Web. 29 Nov. 2016.
- "The Undecidable: Basic Papers on Undecidable Propositions Unsolvable Problems and Computable Functions." *Penn State*. Penn State, n.d. Web. 29 Nov. 2016.
- "Turing Machines." *Lehigh University*. Lehigh University, n.d. Web. 29 Nov. 2016.
- "Turing machine equivalents." *Wikipedia*. N.p., 12 Feb. 2016. Web. 29 Nov. 2016.
- "Universal Turing Machine." *Wikipedia*. Wikimedia Foundation, n.d. Web. 25 Nov. 2016.
- Vollmer, Heribert. *Introduction to Circuit Complexity: A Uniform Approach*. N.p.: n.p., n.d. *Google Books*. Springer Science & Business Media, 17 Apr. 2013. Web. 30 Nov. 2016.
- "What Is a Multi-Track Turing Machine." *Quora*. MarkMonitor Inc, n.d. Web. 29 Nov. 2016.