Great Ideas in Computing

University of Toronto CSC196
Fall 2020

Week 4: October 5-9 (2020)

Week 4: Announcements

Announcements

- This week we have our regular classes on Wednesday and Friday
- Next Monday is Thanksgiving and there is no class.
- Next Wed (October 14), Eyal de Lara will lead a discussion on visualization. I am posting the zoom link on Quercus.
- Quiz 1 will be held Monday, October 19 during the tutorial hour.
- I plan to post Assignment A2 (or the start of A2) this week. Assignment A2 is due October 28 (after the Quiz) but will be helpful for the Quiz.

Week 4 agenda

Agenda

- We will conitnue with Alan Turing's seminal work with regard to what is and what is not computable, and the Turing machine model. I believe this could arguably be called the greatest of the great ideas in computer science that we will discuss. I will elaborate further on Turing's contribution in this regard and provide a very brief description of the Turing machine model.
- I thought we would have time to switch to a more familiar topic, namely search engines. But better to take our time. So next week (week 5) we will strat search engines on Friday.

A pictorial representation of a Turing machine

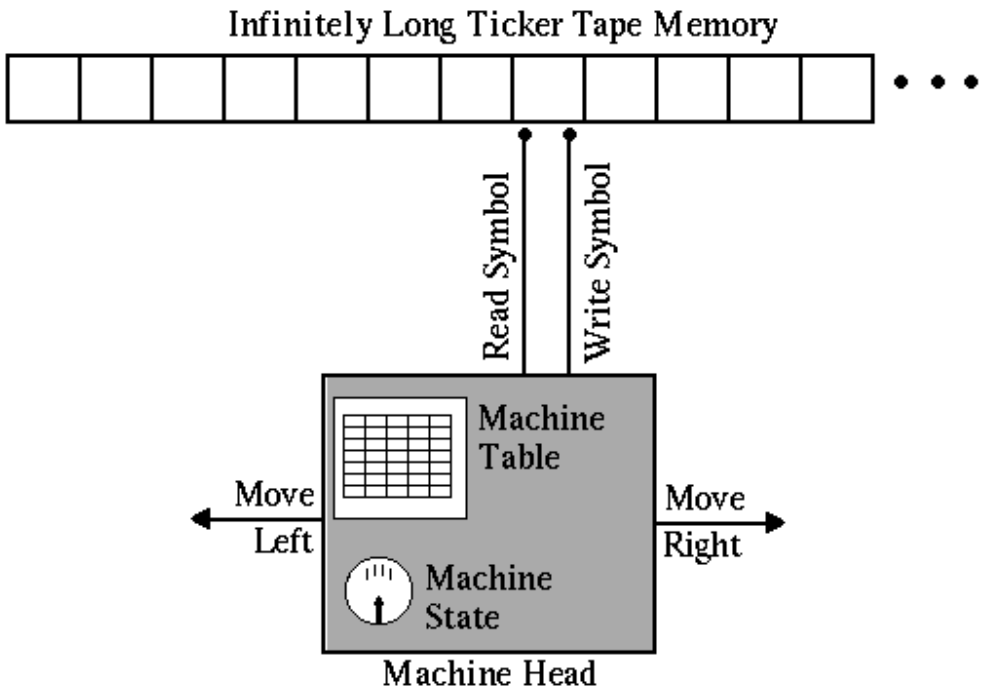

Figure: Figure taken from Michael Dawson "Understanding Cognitive Science"

Comments on Turing's model

- Formally, a Turing machine algorithm is described by the following function $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
  Q is a *finite* set of *states*. Γ is a finite set of symbols
  (e.g., $\Gamma = \{\#, 0, 1, a, b, \ldots\}$ and perhaps $\Sigma = \{0, 1\}$)
- Note: Each δ function is the definition of a single Turing machine; that is, each δ function is the statement of an algorithm.
- We can assume there is a halting state q_{halt} such that the machine halts if it enters state q_{halt}. There is also an initial state q_0.
- We view a Turing machine P as computing a function $f_P : \Sigma^* \rightarrow \Sigma^*$ where $\Sigma \subseteq \Gamma$ where $y = f(x)$ is the string that remains if (and when) the machine halts. There can be other conventions as to interpreting the resulting output y.
- Note that the model is precisely defined as is the concept of a computation step. A *configuration* of a TM is specified by the contents of the tape, the state, and the position of the tape head. A computation of a TM is a sequence of configurations, starting with an initial configuration.
- For decision problems, we can have YES and NO halting states.

A more general Turing machine model

The Turing machine model has been extended to allow separate read (for the input) and write (for the output when computing a function) tapes and any finite number of work tapes. Here is a figure of a nmulti-tape TM (but without separate input and outoput tapes).

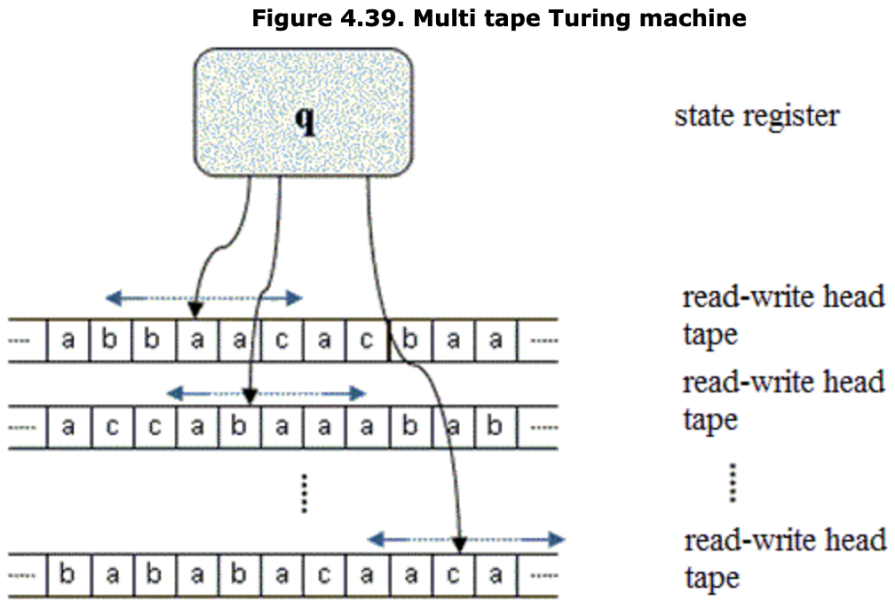

Figure: Figure from the Bela Gyires Infiormatics Curriculum Repository

Some of Turing's seminal results

- Turing showed that there is a Universal Turing machine (UTM) call it U. That is, given an input $p\#x$ the machine interprets the string p as a Turing machine description (i.e. as a state transition function δ) and $x \in \Sigma^*$ is interpreted as the input to the machine P described by p and $f_U(p\#x) = f_P(x)$.
- In modern terms, a UTM is an *interpreter*.
- Turing showed that the **halting problem** is undecideable. That is, there does not exist a fixed TM (call it F) such that for every input p (interpreted as a TM P) will always halt and correctly decide if P halts on some fixed input string (say the single character string 0).
- The undecideability of the halting problem is by a diagonalization argument for those familiar with how you show that the set of real numbers is "larger" (i.e., *an uncountable set*) than the set of rationals or integers (i.e., *countable sets*).

The Entscheidungsproblem

In his seminal paper "On Computable Numbers With an Application to the Entscheidungsproblem" (i.e. decision problem), Turing uses his model and the undecideability of the halting problem, to prove the undecideabiliy of the "Entscheidungsproblem" posed by Hilbert in 1928. (Church provided an independent proof within his formalism.)

Sometimes this is informally stated as "can mathematics be decided ?"

The Entscheidungsproblem question refers to the decideability of validity in prediciate logic which Church and Turing independently resolved in 1936-1937. It would take a little while to fomally define this "Entscheidungsproblem" but here is an example of the kind of question that one wants to answer:
Given a formula such as $\forall x \exists y : y < x$
can we determine if such a formula is always true no matter what what ordered domain x, y and $<$ refer to?
For example, $\forall x, y, z(x < y$ and $y < z$ implies $x \neq z)$ is always true.
But $x < y$ implies $\forall x, y \exists z : x < z < y$ is not true of all ordered domains (e.g., consider the integers)

The extended Church-Turing Thesis

While Church-Turing computability states a formal definition of what is in principle computable, it allows for computable functions of arbitrary complexity.

Indeed it is not difficult to show (again using diagonalization) that for any time bound $T(n)$ there are computable decision problems that require more time (or more memory) than $T(n)$ for sufficiently large values of n where n is the length of the input string. (In complexity analysis we measure complexity as a function of the length of the input and output strings). But why then do we usually say that operations in the von Neumann model take unit time?.

The extended Church-Turing Thesis

While Church-Turing computability states a formal definition of what is in principle computable, it allows for computable functions of arbitrary complexity.

Indeed it is not difficult to show (again using diagonalization) that for any time bound $T(n)$ there are computable decision problems that require more time (or more memory) than $T(n)$ for sufficiently large values of n where n is the length of the input string. (In complexity analysis we measure complexity as a function of the length of the input and output strings). But why then do we usually say that operations in the von Neumann model take unit time?.

The extended Church-Turing thesis states that any function that is "**feasibly computable**" must be computable by a Turing machine within time $p(n)$ where $p()$ is a polynomial. This extended thesis is stated with regard to classical computers and not necessarily quantum computers.

Extended Church-Turing thesis continued

Informal theorem: Any "**reasonable"** classical computation model $\mathcal{M}$ can be simulated by a one tape or multi-tape Turing machine so that if say f is computable in time $T(n)$ on $\mathcal{M}$ then f is computable on a Turing machine in time $p_{\mathcal{M}}(T(n))$ for some fixed polynomial $p_{\mathcal{M}}$.

Using a mutli-tape TM, for most models, $p_{\mathcal{M}}(m)$ is $O(m^2)$ or $O(m^3)$. For example, if $T(n)$ is n^2 and $p_{\mathcal{M}}(m)$ is m^3 then $p_{\mathcal{M}}(T(n)$ is n^6.

What is **"not reasonable"**?

Extended Church-Turing thesis continued

Informal theorem: Any "**reasonable**" classical computation model $\mathcal{M}$ can be simulated by a one tape or multi-tape Turing machine so that if say f is computable in time $T(n)$ on $\mathcal{M}$ then f is computable on a Turing machine in time $p_{\mathcal{M}}(T(n))$ for some fixed polynomial $p_{\mathcal{M}}$.

Using a mutli-tape TM, for most models, $p_{\mathcal{M}}(m)$ is $O(m^2)$ or $O(m^3)$. For example, if $T(n)$ is n^2 and $p_{\mathcal{M}}(m)$ is m^3 then $p_{\mathcal{M}}(T(n)$ is n^6.

What is **"not reasonable"**?

Consider having unit time operations $+, -, *, \div$ where $\div$ means integer division; that is, dividing a by b results in the remainder. Now consider repeated squaring of $2, 2^2 = 4, 4^2 = 16, \ldots, 2^{(2^n)}$. That is, in n multiplications, we can construct an integer whose binary representation has 2^n bits.

It turns out that with such a model we can factor integers in polynomial time. **BUT** this is not reasonable as we are doing classical operations on exponentially long strings in unit time which is not reasonable.