Great Ideas in Computing

University of Toronto CSC196
Fall 2020

Week 3: September 28 - October 2 (2020)

Week 3: Agenda

- We started this week with our guest speaker, Professor Henry Yuen who led a discussion on *quantum computing*
- I have often started my previous SCI 199 courses by discussing 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 on Turing's contribution in this regard but at this time I won't dwell on the details of the Turing machine model.
- Then (probably next week) I will switch to a more familiar topic, namely search engines. I consider search engines as one of three "killer applications" that has made computing so pervasive and in fact, a necessity for almost everyone. What are my other two killer applications?

Week 3: Agenda

- We started this week with our guest speaker, Professor Henry Yuen who led a discussion on *quantum computing*
- I have often started my previous SCI 199 courses by discussing 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 on Turing's contribution in this regard but at this time I won't dwell on the details of the Turing machine model.
- Then (probably next week) I will switch to a more familiar topic, namely search engines. I consider search engines as one of three "killer applications" that has made computing so pervasive and in fact, a necessity for almost everyone. What are my other two killer applications?

  Email and Social Networks

Week 3: Agenda

- We started this week with our guest speaker, Professor Henry Yuen who led a discussion on *quantum computing*
- I have often started my previous SCI 199 courses by discussing 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 on Turing's contribution in this regard but at this time I won't dwell on the details of the Turing machine model.
- Then (probably next week) I will switch to a more familiar topic, namely search engines. I consider search engines as one of three "killer applications" that has made computing so pervasive and in fact, a necessity for almost everyone. What are my other two killer applications?

  Email and Social Networks

  What would be your candidates for the most influential killer applications?

Announcements

- For general information about writing resources at U of T, students can started at the home page of the site Writing at The University of Toronto: https://writing.utoronto.ca.
- I would like to call your attention to the following site concerning the Toronto Experimental Economics Lab (TEEL). This seems interesting. I am not sure if I will discuss algorithmic game theory and mechanism design in our course, but we do have a course CSC304 on that topic.

  Volunteers do get paid.
  'Would there be interest in a 5 minute presentation by someone from the TEEL?
- Schedule for remaining guests:
  - October 14 Eyal deLara virtualization
  - October 28 Roger Grosse ML and deep learning
  - November 18 Alexandar Nikolov differential privacy

A few more annoucements

- I have posted A1 on Markus so that the asasignment should be submitted there. Please let me know if you have any issues regarding submitting.
- I have changed the late policy to be 5% penalty for each 24 hours up to 96 hours. A1 is due October 7 at 11 AM.
- I have posted Henry Yuan's slides on the web page.
- I think we will continue to use Zoom for guest classes unless the guest wants to use Bb Collaborate. I gave out the link for the recording and we are hoping to post it on Quercus.

Some comments on Professor Yuan's discussion on quantum computing

- As was stated there can be a lot of hype about being on the verge of large scale quantum computing "which will change everything".
- Currently, quantum computing is far from being "large scale".
- The history of computing has been one of rapid advancement. While there are reasons to be skeptical about the future of large scale quantum computing, one should keep in mind the reflection about the early days of computing being based on vacuum tubes and then being revolutionized by the transistor.
- Killer applications (such as factoring) have already led to increased research in quantum computing.
  Aside: Peter Shor indicates the motivation provided by a paper by Dan Simon in his UT/DCS PHD supervised by Charles Rackoff (recently retired), one of the main researchers in cryptography.
- We know that searching for an item in an unordered list of n items requires n accesses and Grover's quantum algorithm uses only $O(\sqrt{n})$ accesses which is a *provable* speedup in contrast to the conjectured almost exponential speedup by quantum computing for factoring.

Computer Science as a mathematical science

David Hilbert was one of the great mathematicians of the late 19th and early 20th centuries. He asked the following question in **1900** known as Hilberts' 10^{th} problem:

"Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers"

Here is a more familiar way to ask this question:

Given a polynomial $P(x_1, \ldots, x_n)$ with integer coefficients in many variables, decide if P has an integer root. That is, do there exist integers $i_1, \ldots, i_n$ such that $P(i_1, \ldots, i_n) = 0$?

As an example, $P(x) = x - 2$ clearly has an integer root whereeas $P(x) = x^2 - 2$ does not have an integer (or rational) root.

What is computable? What is decideable?

Hilbert's question was essentially to ask if there is an algorithm that could decide whether or not a given multivariate polynomial has an integer root. Hilbert didn't mention the words "algorithm" or "computer" but he did articulate the need to solve the problem in a finite number of "steps".

Hilbert believed there was such a decision procedure but did not formalize what it meant to say that a problem solution is *computable*.

Terminology: If the problem is a decision problem (i.e., where the solution is to output YES or NO) then we usually say decideable rather than computable.

Following a series of intermediate results over 21 years, in 1970 Matiyasevich gave the first proof that Hilbert's 10th problem was undecideable (in a precise sense we will next discuss).

Note: The problem is decideable for polynomials in one variable.

Is there a precise definition for the meaning of "decideable"

We have studied tha von Neumann model as a model of computation but we never gsave a precise definition but more or less relied on our prior knowledge of how we think computers work. And we didn't give a definition for what is an *algorithm*.

Computers are continually getting faster and have larger memories so must our concept of what is computable also be constantly changing? Could Hilbert's problem become decideable tomorrow?

We also briefly touched upon the complexity of operations with respect to the data structures for the dictionary data type. Must the complexity of operations and the complexity of algorithms also change constantly?

This raises a fundamental question: Is there an ultimate precise model of computation with respect to which we would then have a precise meaning of a computable function? Or must we continually be changing our understanding of what is and what is not computable?

Is there a precise definition for the meaning of computable (decideable) continued

High level models such as the von Neumann model provide a good intuition for what we have in mind when we say a function f is decideable. **But we really need a precise mathematical model if we want to prove mathematical results.**

Independently in 1936, Alonzo Church and Alan Turing published formal definitions for what it meant to be computable. These papares were very influential for von Neumanns model which comes about 10 years later.

Church's definition was based on a formalism in logic called the lambda calculus where one starts with some basic functions and then :/Over m applies ways to compose new functions from existing functions.

Alan Turing proposed a precise model of computation which we will (for now) only briefly describe. Turing also went on to show that these very diffefrent models are provably equivalent in the sense that they result in the same set of computable functions.

But are there other models?

For a number of years other models were considered and all turn out to be equivalent (and sometimes weaker) than the Church-Turing models.

This led to the following Church-Turing hypothesis. Every plausible model of computation is equivalent to (or weaker than) Turing's very basic computational model. This is not to say that Turing machines are as easy to program or will lead to the same complexity analysis. **But the meaning of computable does not change.**

In particular, what about quantum computing?

But are there other models?

For a number of years other models were considered and all turn out to be equivalent (and sometimes weaker) than the Church-Turing models.

This led to the following Church-Turing hypothesis. Every plausible model of computation is equivalent to (or weaker than) Turing's very basic computational model. This is not to say that Turing machines are as easy to program or will lead to the same complexity analysis. **But the meaning of computable does not change.**

In particular, what about quantum computing? It could very well be that quantum computing will substantially change our sense of what is "efficiently computable" but it does not enlarge the meaning of "computable".

We ended Friday's meeting and Week 3 on nthis slide. WE coontinue next Wedneday and Friday with the Turing model.