Computation and Discrete Mathematics

Klaus Sutner
Carnegie Mellon University
Spring 2021

Dramatis Personae 2

- Prof:
  Klaus Sutner, sutner@cs.cmu.edu
- TA:
  XXXX @andrew.cmu.edu
- Course secretary:
  Rosie Battenfelder, rosemary@cs.cmu.edu

The Covid Mess 3

Under normal circumstances, this section would be simply an enumeration of well-tested policies and approaches.

Alas, things are a bit in flux right now–we may have to adjust on the fly if new problems arise.

I have made a good number of adjustments, based on my observations over the last year, but I'm not sure that things could not be tweaked in one way or another.

So, take all of this with a grain of salt, I'll keep you updated if things do change.

Bureaucracy 4

- The usual assessment:
  - homeworks 40%
  - midterm (take-home) 30%
  - final (take-home) 30%
- HW is critical for CDM, for your grade (summative) but more importantly for feedback and understanding (formative). Make sure to invest significant effort.
- There are no makeups; if you miss some assessment you can sit for an oral exam (I am not a great supporter of the currently popular high school approach to university education).

Bureaucracy II 5

- You have a total of 6 (six) late days at your disposal; use prudently.
- A late day is a discrete atom, with no smaller parts. Mention lateness in the header of your HW.
- If you run into problems, talk to us early, don't wait till the end of the semester when a train wreck is imminent.

Preserving TA Sanity

- Typeset your homework solutions and submit pdf on Gradescope (more on this on Piazza).
- If you have extensive conversations with other students about a HW, mention them as "collaborators" in your submission.
- If you use a computer program in your homework, make sure to reference it properly (but do **not** hand in 50 pages of code).

Cooperation and Limits

- You are strongly encouraged to talk about the course material to each other and the course staff.
- This includes discussions of homework problems.
- However, even after ample consultation, the work you submit must be written entirely by yourself.
- List all your "consultants" on the first page of your homework.
- Cite other resources used (websites, research papers, . . .)
- To avoid problems with originality, do not take notes when discussing homework problems.

More Limits

And, of course, all the official university policies apply.

http://www.cmu.edu/academic-integrity/index.html

Computing

Some HW problems are best attacked by some experimental computation. Without meaning to start a war, here is an environment that I have found quite useful:

- OS: fedora/popos, window manager: i3/mutter, shell: fish
- Mathematica, SAGE, C++, LEAN, . . .
- emacs, auctex, cdlatex, latexrun, tex-live

For desktops a tiling wm seems superior, for laptops the answer is far less clear. Experiment.

Web and Communication

Course Website: CDM

Piazza piazza

Asking Piazza Questions

Always use descriptive titles:

> HW 3, Q 2: why is f primitive recursive?

> Lect 5, sld 15: gap in proof of theorem

Provide all the necessary information, include links whenever appropriate. If the question is ill-posed, the likelihood of a useful answer diminishes exponentially.

Also, read other posts first, someone else may well have asked the same question.

More Piazza

Our Piazza has a label category Notes.

Use this tag to point out problems with lecture slides and lecture notes.

This is not just about a typos or plain errors–I also want to hear if a slide is misleading, or difficult to read and understand, another example is needed, . . . Ideally, propose an improvement. But at the least let me know that there is a problem.

Email

Some of you may still remember email (a medieval communication tool involving pigeons, predating social networks by thousands of years). If you decide to get in touch with me via email, use

- Subject line:

  `[CDM] will miss midterm`

or some such. I have hacked `emacs vm` and filter rather aggressively, make sure to have the `[CDM]` tag.

Course Material

- There is no text book.
- There is a lot of material (slides, lecture notes, papers) at CDM. Familiarize yourself with the site, now.
- Zoom lectures are less than optimal, so I will ask you to skim the slides ahead of lecture.
- Hopefully the small class size will make some amount of interaction possible during lecture.

Sources in General

At this point, there is a large amount of high-quality material on the web.

Make sure you know where to look.

- Google.
- Google scholar.
- Course notes, online courses, blogs.
- DBLP, ECCC, arXiv.
- Conferences (FOCS,STOC,SODA,CCC,ICALP).

Wisdom

> The utterly pure theory of mathematical proof and the utterly technological theory of machine computation are at bottom one, and the basic insights of each are insights of the other.

This is a comment by W. V. O. Quine in his book *The Ways of Paradox and Other Essays*, chapter "On The Application of Modern Logic".

Central CDM Axiom

> The single-most important event in mathematics in the 20th century is the development of the digital computer.

Yes, yes, this is far less polite than Quine. In fact, it is an outrage, a vile affront against common scholarly standards of circumlocution, decency and decorum. But bear with me.

Scope

To be clear, we are not only suggesting that various surrounding theories (such as complexity theory) are important, it's also the actual physical devices that matter:

Without these, the theory apparatus would likely not have been developed and/or would have been purely academic.

Using Computers

Here is a list of applications of computers (the machines, not just the theory) in mathematics, in strictly decreasing order of acceptance and frequency of use.

- Number Crunching
- Symbolic Computation
- Mathematical Knowledge Management (MKM)
- Theorem Proving

Fairly Standard

- Number Crunching

  Solving complicated differential equations; optimization problems. Historically the first major application; even though real arithmetic is difficult on digital machines.

- Symbolic Computation

  Directly manipulate symbolically presented entities (computer algebra, SAT solvers, SMT solvers, model checkers).

  - large, complex computations
  - example/counterexample generation
  - integrated computational environments

Not So Much

- Knowledge Management

  A global mathematical library would be some 10^8 pages. Some small but growing part of this is available in digital form on the web. Some even smaller part is indexed and searchable (semantic markup). Some yet smaller part is validated.

- Proof Checking, Theorem Proving

  On some occasions, proof assistants and automatic theorem provers are helpful in finding (parts of) a proof. Better at verification (proof checkers) than search. Alas, they all require substantial technical skill on the side of the user.

Why Would CS Care?

Tony Hoare's inaugural address in Oxford 1977:

- Computers are mathematical machines.
- Computer programs are mathematical expressions.
- A programming language is a mathematical theory.
- Programming is a mathematical activity.

Many have tried to disagree, but there really is no reasonable alternative. In particular the idea that classical engineering ideas can solve fundamental problems in CS is laughable.

The Critical Link

Mathematical logic is classically divided into computability theory, proof theory, set theory and model theory. The first two are critical for CS.

math	cs	cs foundations
proof	*program*	*term*
proposition	*specification*	*type*

The idea of thinking of programs as certain kinds of proof is extremely helpful when it comes to correctness considerations. Alas, we will not pursue the type theory angle here.

Seriously?

Here are two recent standout developments that strongly suggest the connection between math and computing will open new doors.

- A link between classical undecidability and quantum protocols.
- LEAN, an increasingly popular theorem prover

MIP$*$ = RE

MIP refers to a multi-interactive-prover protocol (the computationally powerful provers try to convince the weak verifier). The star means the provers have entangled qubits. RE is the collection of all recursively enumerable sets–which includes the undecidable Halting Set.

Alas, the provers require a rather large number of entangled qubits. Don't expect such provers to be implementable in actual physics.

Interestingly, the work also answered two other major questions:

- Tsirelson's problem concerning models of particle entanglement.
- Connes' embedding conjecture.

LEAN

LEAN is a state-of-the-art theorem prover, written mostly by Leo de Moura at MS Research. Ironically, it was originally intended for program verification.

Like all its competitors, it requires a huge library of formalized mathematics to become a useful tool; building such a library is a sysiphean undertaking that is many years from completion.

The good news is that theorem provers based on sophisticated type systems have attracted the attention of a Fields medalist (the late Vladimir Voevodsky),

It seems that LEAN is becoming the de facto standard for mathematicians interested in theorem proving mathlib.

A Brief History of Proofs

Not really, just a caricature that homes in on a view important points.

Standard math proofs are no more than high-level sketches. They are directed at smart, highly trained experts, who spend years and even decades on becoming familiar with a particular field and style of exposition. In particular, they have learned to use a combination of intuition and formal skills to fill in gaps and interpret ambiguous assertions.

A prover/proof checker is a completely mindless and purely mechanical device, a "persistent plodder" (Hao Wang). It will do exactly what it is programmed to do, no more and no less. If there is even the slightest flaw in the alleged proof, the verification attempt will fail (with luck by pointing out the place where the argument is deficient). Also, the mechanical proofs may well be near incomprehensible.

The Good Old Days

Aka Ancient Mathematical History. For our purposes, let's say that's everything up to 1800.

Up to this point, mathematics was doing just fine. There were fierce debates, but no one doubted the fundamental soundness of mathematical reasoning. Practitioners might and did blunder; other than that, results were good in perpetuity.

E.g., Euclid's old argument for the infinitude of primes is unassailable (unless you are a dyed in the wool constructivist). And his system of geometry was obviously correct.

Approaching the Abyss

But things started to unravel a bit in the 19th century. Here is a most revealing quote in a letter to A. von Humboldt:

> If Gauss says he has proved something, it seems very probable to me; if Cauchy says so, it is about as likely as not; if Dirichlet says so, it is certain.
>
> C.G.J. Jacobi

Jacobi was no slouch but one of the experts from the last slide; if he criticizes a proof it is because there are real, serious and difficult-to-fix problems, not just some superficial lack of understanding.

Foundations

How does one construct solid proofs? What are the underlying mathematical foundations? Up to the middle of the 19th century, basically no one cared.

It was standard to follow in Kant's footsteps (who, like Aristotle, knew little math) and adopt a theory that:

- spatial intuition is a given of human cognition, and gives rise to geometry,
- intuition of time is similarly given, and leads to arithmetic.

Even Hamilton (as in Hamiltonian systems and quaternions) tried to formulate such a model in 1853.

Intuition versus Abstraction

The problem with intuition-based reasoning is that it starts to fail when the subject of discourse becomes increasingly abstract and sophisticated (some would say: abstruse). Ordinary experience and every-day notions start to lose traction. The 19th century saw a lot of that.

To be clear: even today not everyone participates in the quest for absolute precision. For example, physics super-star Steven Weinberg writes in a book on quantum field theory

> . . . there are parts of this book that will bring tears to the eyes of the mathematically inclined reader.

In physics, this is probably a good thing that helps the field along. In CS, it would more likely be a disaster.

Euler, an Example

Leonhard Euler had the amazing ability to concoct arguments that were eminently plausible, and led to correct results, but were exceedingly difficult to justify in the modern sense. Here is an example:

> **Problem:** Find a way to calculate e^x for real x.

Wlog $x > 0$. For x reasonably small we can write

$$e^x = 1 + x + error$$

with the error term being small. Alas, we have no idea what exactly the error is (duh).

Infinitesimals

Euler considers an infinitesimal $\delta > 0$. This simplifies matters greatly:

$$e^\delta = 1 + \delta$$

It is justified by Leibniz's lex homogeneorum transcendentalis, the transcendental law of homogeneity: proof of the law is by higher authority (or nonstandard models of arithmetic 250 years later).

Once we accept the last identity, we can calculate happily:

Finale Furioso

$$
\begin{aligned}
e^x &= (e^\delta)^{x/\delta} \\
&= (1+\delta)^{x/\delta} \\
&= 1 + \binom{x/\delta}{1}\delta + \binom{x/\delta}{2}\delta^2 + \ldots \\
&= 1 + x + 1/2\, x(x-\delta) + \ldots \\
&= 1 + x + 1/2\, x^2 + \ldots \\
&= \sum_{i \geq 0} x^i/i!
\end{aligned}
$$

Headache?

The result is perfectly correct. But the argument . . . oy vey.

Incidentally, Euler also had an ingenious argument to show that

$$1 + 2 + 3 + \ldots + n + \ldots = -\frac{1}{12}$$

No, that's not nuts.

Look up ζ function regularization and analytic continuations.

Food for Thought

Exercise

Find all the places where Euler's reasoning is dubious from a modern day perspective.

Exercise (Very Hard)

Fix all the problems with Euler's argument.

Exercise

Figure out why it makes sense to claim that $1 + 2 + 3 + \ldots = -1/12$. *Similarly* $1 + 1 + 1 + \ldots = -1/2$.

The Antidote

Apparently, the only reliable (and somewhat unpalatable) solution to this problem of rigor is to be exceedingly formal and precise in all arguments. At least four frameworks emerged that appear to be helpful in this enterprise (perhaps in combination):

- Logic and Formalization (Boole, Frege, Peano)
- Axiomatization (Peano, Dedekind, Hilbert)
- Set theory (Dedekind, Cantor, Frege, Zermelo, Fraenkel)
- Type theory (Russell, Howard-Curry, Church, Per-Löf, Voevodsky)

Quoi?

The first two merged more or less into the notion of a formal system: a suitable logic, that makes it possible to formalize appropriate axioms and rules of inference (first-order logic is the standard).

Under Bourbaki, set theory developed into the reference implementation, the de facto gold standard that everybody relies on (though most practitioners would have a hard time explaining ZF).

Type theory was off to a great start under Russell and Whitehead, but got clobbered by set theory. However, at present type theory is arguably more important in CS, and as suggested by MKM and theorem proving, might at some point also become dominant in math.

Hilbert's Program

In the 1920s, partially in response to paradoxes and intuitionistic lunacy, David Hilbert proposed a program to salvage all of mathematics. In a nutshell:

> Formalize mathematics and concoct a finite set of axioms that are strong enough to prove all theorems of mathematics (completeness) and show that the system is free from contradictions (consistency); by strictly finitary means.
>
> Also show that statements about "ideal objects" can be proven in the system, without using ideal objects.

Perfect for CS

Strictly finitary means that every single logical operation has to be entirely mechanical and, in a sense, trivial. In particular, they could easily be carried out by a computer (some would say that's the only thing a computer can do). In particular it must be decidable whether a given argument is a valid proof.

The part about ideal objects means that reasoning about infinite objects is fine as long as it can be justified in a finitary way.

Initially good progress was made

- completeness of propositional logic (Ackermann 1928),
- completeness of predicate logic (Gödel 1930).

Gödel

But then, in 1931, Kurt Gödel drops a bombshell: any formal mathematical system of the kind Hilbert had in mind is necessarily incomplete or inconsistent. So in any consistent system some true statements cannot be proven.

In particular consistency itself (i.e. lack of internal contradictions) is most elusive: e.g. one cannot prove consistency of arithmetic in arithmetic.

Interestingly, Gödel's argument is based on a version of the old Epimenides paradox, exploiting self-reference:

> This sentence is false.

Note that this type of paradox is quite different from the purely logical paradox of Russell (see below) and far more opaque.

Computability

Though some parts of Hilbert's program were irreparably damaged by Gödel's result, in many ways things just started to become really interesting.

> Thus the notion "computable" is in a certain sense "absolute," while almost all metamathematical notions otherwise known (for example, provable, definable, and so on) quite essentially depend upon the system adopted.
>
> K. Gödel, 1936

There is a clear connection between incompleteness and unsolvability, so computability is a rather central notion in mathematics.

Entscheidungsproblem

> The Entscheidungsproblem is solved when one knows a procedure by which one can decide in a finite number of operations whether a given logical expression is generally valid or is satisfiable. The solution of the Entscheidungsproblem is of fundamental importance for the theory of all fields, the theorems of which are at all capable of logical development from finitely many axioms.
>
> D. Hilbert, W. Ackermann
> Grundzüge der theoretischen Logik, 1928

In modern terminology: find a *decision algorithm* for statements of mathematics (or at least some part like arithmetic, group theory, ...).

Naysayers

In the interest of fairness, we point out again that the whole premiss about the importance of computation and computers in math (and even in CS) is certainly not without its detractors.

In fact, it drives some people right up the wall.

Here is an excerpt from a paper written in 1991 by a propellerhead who shall go unnamed and unmentioned.

Computers Bad

> Lastly, and most dastardly, there is the much ballyhooed extension of the notion of mathematical proof, far beyond the classical Greek paradigm of of axioms and rules of inference, to include so-called computer-proofs, whose non-surveyability by human beings demands appeals to faith inimical to the enterprise of science, and yet allegedly yields results which are otherwise of necessity far beyond the powers of mortal man to obtain.
>
> A mediocre mathematician with a computer might be able to simulate the creative powers of a top notch mathematician with pencil and paper.

Computers Very Bad

OK, one more quote from the same guy.

> Admitting the computer shenanigans of Appel and Haken to the ranks of mathematics would only leave us intellectually unsatisfied.

As my old high school philosophy teacher said, after some student complained about not being satisfied with a lecture: That issue you will have to take into your own hands.

The Catch?

So where is the catch? If it is true that the math–computation connection is so utterly useful and compelling, why is it that in 2021 the standard approach to math is not a trip to the computational universe? Why is there a lag of whole century?

Of course, old habits die hard, so one should expect some amount of resistance. This is particularly relevant here where some of the new ideas come from a different, and often ridiculed field.

And then there is a famous problem pointed out by a well-known American philosopher of the 20th century.

Infinite Wisdom

> In theory there is no difference between theory and practice.
>
> In practice there is.
>
> Yogi Berra

A Warning

> The standard of correctness and completeness necessary to get a computer program to work at all is a couple of orders of magnitude higher than the mathematical community's standard of valid proofs.
>
> Bill Thurston, Notices AMS, 1994

Fields Medal 1982, used computers extensively in his seminal work on low-dimensional topology. Supposedly taught himself to visualize 4-dim objects.

To be honest, Thurston had ulterior motives to make this statement, but that does not detract from its fundamental correctness.

But the Future is Golden

> I expect that digital computing machines will eventually stimulate a considerable interest in symbolic logic . . . The language in which one communicates with these machines . . . forms a sort of symbolic logic.
>
> Alan Turing

> The Algorithm's coming-of-age as the new language of science promises to be the most disruptive scientific development since quantum mechanics.
>
> Bernard Chazelle

> Generally, computer science, that no-nonsense child of logic, will exert growing influence on our thinking about the languages by which we express our vision of mathematics.
>
> Yuri Manin

Battleplan

- Brief introduction to computability and complexity.
- Selected topics in discrete math:
  - Transition systems
  - Groups and finite fields

Alas, discrete math is far too broad to fit into a single semester, so take this as a proof of concept.

Reading

- Take a look at the notes on primitive recursive functions on the web.
- Don't worry about technical details, just get a general idea.
- Ask questions on Thursday (or on Piazza if time-critical).