

How will we do mathematics in 2030 ?

Michael R. Douglas

CMSA / Harvard University and Stony Brook University

CERN Colloquium, Sep 30, 2021

Abstract

We make the case that over the coming decade, computational technology and computer assisted reasoning will become far more widely used in the mathematical sciences.

Extended Abstract

We make the case that over the coming decade, computer assisted reasoning will become far more widely used in the mathematical sciences. This includes interactive and automatic theorem verification, symbolic algebra, and emerging technologies such as formal knowledge repositories, semantic search and intelligent textbooks. After a short review of the state of the art, we survey directions where we expect progress, such as mathematical search and formal abstracts, developments in computational mathematics, integration of computation into textbooks, and organizing and verifying large calculations and proofs. For each we try to identify the barriers and potential solutions.

Computers and the internet have had a large and growing impact on many human activities. Many believe that machine learning and AI will dramatically expand this impact.

In this talk we focus on the mathematical sciences, broadly defined to include not just pure and applied mathematics and statistics, but also much of computer science, and the theoretical and computational branches of physics, chemistry and biology.

Our goal will be to identify opportunities for significant advances in how we discover, communicate and teach knowledge in these fields, which are feasible over the coming decade.

We will do this by identifying lines of research which address clear needs, which are being pursued now, which have interesting results and are making progress, and then extrapolating this progress.

Computers and the internet have had a large and growing impact on many human activities. Many believe that machine learning and AI will dramatically expand this impact.

In this talk we focus on the mathematical sciences, broadly defined to include not just pure and applied mathematics and statistics, but also much of computer science, and the theoretical and computational branches of physics, chemistry and biology.

Our goal will be to identify opportunities for significant advances in how we discover, communicate and teach knowledge in these fields, which are feasible over the coming decade.

We will do this by identifying lines of research which address clear needs, which are being pursued now, which have interesting results and are making progress, and then extrapolating this progress.

Computers and the internet have had a large and growing impact on many human activities. Many believe that machine learning and AI will dramatically expand this impact.

In this talk we focus on the mathematical sciences, broadly defined to include not just pure and applied mathematics and statistics, but also much of computer science, and the theoretical and computational branches of physics, chemistry and biology.

Our goal will be to identify opportunities for significant advances in how we discover, communicate and teach knowledge in these fields, which are feasible over the coming decade.

We will do this by identifying lines of research which address clear needs, which are being pursued now, which have interesting results and are making progress, and then extrapolating this progress.

Computers and the internet have had a large and growing impact on many human activities. Many believe that machine learning and AI will dramatically expand this impact.

In this talk we focus on the mathematical sciences, broadly defined to include not just pure and applied mathematics and statistics, but also much of computer science, and the theoretical and computational branches of physics, chemistry and biology.

Our goal will be to identify opportunities for significant advances in how we discover, communicate and teach knowledge in these fields, which are feasible over the coming decade.

We will do this by identifying lines of research which address clear needs, which are being pursued now, which have interesting results and are making progress, and then extrapolating this progress.

The first electronic computers were built at the end of World War II, and they were first used for numerical calculations in scientific and military applications. Soon the pioneers of computer science saw their far broader potential:

- Information storage and retrieval (Vannevar Bush)
- Control systems (many people, let's say Norbert Wiener)
- Artificial intelligence (Newell and Simon, McCarthy, Minsky)
- Natural computing (Rosenblatt, Ulam and von Neumann, Holland)

In mathematical research, the growing availability of computation led to a steadily growing interest in numerical methods and simulation. This has had a huge impact on the mathematical sciences.

AI researchers proposed visionary ideas such as automated theorem proving. Early attempts were based on tree search, which in general requires exponential time and by itself can only solve small problems. But as part of their research, they developed broadly useful technologies for symbol manipulation, leading for example to symbolic algebra systems.

The first electronic computers were built at the end of World War II, and they were first used for numerical calculations in scientific and military applications. Soon the pioneers of computer science saw their far broader potential:

- Information storage and retrieval (Vannevar Bush)
- Control systems (many people, let's say Norbert Wiener)
- Artificial intelligence (Newell and Simon, McCarthy, Minsky)
- Natural computing (Rosenblatt, Ulam and von Neumann, Holland)

In mathematical research, the growing availability of computation led to a steadily growing interest in numerical methods and simulation. This has had a huge impact on the mathematical sciences.

AI researchers proposed visionary ideas such as automated theorem proving. Early attempts were based on tree search, which in general requires exponential time and by itself can only solve small problems. But as part of their research, they developed broadly useful technologies for symbol manipulation, leading for example to symbolic algebra systems.

The first electronic computers were built at the end of World War II, and they were first used for numerical calculations in scientific and military applications. Soon the pioneers of computer science saw their far broader potential:

- Information storage and retrieval (Vannevar Bush)
- Control systems (many people, let's say Norbert Wiener)
- Artificial intelligence (Newell and Simon, McCarthy, Minsky)
- Natural computing (Rosenblatt, Ulam and von Neumann, Holland)

In mathematical research, the growing availability of computation led to a steadily growing interest in numerical methods and simulation. This has had a huge impact on the mathematical sciences.

AI researchers proposed visionary ideas such as automated theorem proving. Early attempts were based on tree search, which in general requires exponential time and by itself can only solve small problems. But as part of their research, they developed broadly useful technologies for symbol manipulation, leading for example to symbolic algebra systems.

The next technological leap was the Internet. This dramatically sped up the dissemination of results. In my own field of theoretical physics, we went from paper preprints in the 80's to the arXiv in the early 90's.

The “second superstring revolution” was begun in 1993 by Ashoke Sen, working almost alone in India. Before the internet, important discoveries were quickly taken over by groups working in a few dominant centers. But with the internet, Sen and the many other researchers spread around the world could stay competitive. Unlike the “first superstring revolution” in the 80's, this time there was no single dominant center of research.

Arguably the greatest success story of this type is Wikipedia. In the past, large collaborative knowledge projects such as encyclopedias required a great deal of organization, and centralized control of the editing process. While Wikipedia still has editors and a hierarchical organization, it requires far fewer editors than anyone predicted. Wikipedia is a valuable resource for mathematical scientists, but by no means a panacea.

The next technological leap was the Internet. This dramatically sped up the dissemination of results. In my own field of theoretical physics, we went from paper preprints in the 80's to the arXiv in the early 90's.

The “second superstring revolution” was begun in 1993 by Ashoke Sen, working almost alone in India. Before the internet, important discoveries were quickly taken over by groups working in a few dominant centers. But with the internet, Sen and the many other researchers spread around the world could stay competitive. Unlike the “first superstring revolution” in the 80's, this time there was no single dominant center of research.

Arguably the greatest success story of this type is Wikipedia. In the past, large collaborative knowledge projects such as encyclopedias required a great deal of organization, and centralized control of the editing process. While Wikipedia still has editors and a hierarchical organization, it requires far fewer editors than anyone predicted. Wikipedia is a valuable resource for mathematical scientists, but by no means a panacea.

The next technological leap was the Internet. This dramatically sped up the dissemination of results. In my own field of theoretical physics, we went from paper preprints in the 80's to the arXiv in the early 90's.

The “second superstring revolution” was begun in 1993 by Ashoke Sen, working almost alone in India. Before the internet, important discoveries were quickly taken over by groups working in a few dominant centers. But with the internet, Sen and the many other researchers spread around the world could stay competitive. Unlike the “first superstring revolution” in the 80's, this time there was no single dominant center of research.

Arguably the greatest success story of this type is Wikipedia. In the past, large collaborative knowledge projects such as encyclopedias required a great deal of organization, and centralized control of the editing process. While Wikipedia still has editors and a hierarchical organization, it requires far fewer editors than anyone predicted. Wikipedia is a valuable resource for mathematical scientists, but by no means a panacea.

Many people would agree that we are in the midst of a third technological leap, powered by **machine learning**. ML has led to dramatic improvements in the ability of computers to recognize and classify patterns, to translate between languages, to play games of strategy, to extract information from documents, and to carry out tasks without requiring explicit programming. Its scope is still expanding.

ML is already a core technology for firms like Google, Apple, Facebook *etc.* and they have established groups whose total research staff numbers in the thousands. ML is gradually being adopted by businesses of all sizes, academic research groups, government *etc.*

How can ML be used by mathematical scientists? Of course ML is itself a mathematical concept and is studied by statisticians, applied mathematicians and other mathematical scientists. But it can also be used to augment the human capability to do more general research, by helping to recognize patterns, by finding and organizing relevant data, documents and code, by helping to write and verify code and mathematical proofs, in education, and in other ways.

Many people would agree that we are in the midst of a third technological leap, powered by **machine learning**. ML has led to dramatic improvements in the ability of computers to recognize and classify patterns, to translate between languages, to play games of strategy, to extract information from documents, and to carry out tasks without requiring explicit programming. Its scope is still expanding.

ML is already a core technology for firms like Google, Apple, Facebook *etc.* and they have established groups whose total research staff numbers in the thousands. ML is gradually being adopted by businesses of all sizes, academic research groups, government *etc.*

How can ML be used by mathematical scientists? Of course ML is itself a mathematical concept and is studied by statisticians, applied mathematicians and other mathematical scientists. But it can also be used to augment the human capability to do more general research, by helping to recognize patterns, by finding and organizing relevant data, documents and code, by helping to write and verify code and mathematical proofs, in education, and in other ways.

Many people would agree that we are in the midst of a third technological leap, powered by **machine learning**. ML has led to dramatic improvements in the ability of computers to recognize and classify patterns, to translate between languages, to play games of strategy, to extract information from documents, and to carry out tasks without requiring explicit programming. Its scope is still expanding.

ML is already a core technology for firms like Google, Apple, Facebook *etc.* and they have established groups whose total research staff numbers in the thousands. ML is gradually being adopted by businesses of all sizes, academic research groups, government *etc.*

How can ML be used by mathematical scientists? Of course ML is itself a mathematical concept and is studied by statisticians, applied mathematicians and other mathematical scientists. But it can also be used to augment the human capability to do more general research, by helping to recognize patterns, by finding and organizing relevant data, documents and code, by helping to write and verify code and mathematical proofs, in education, and in other ways.

Established media and tools

Let's briefly survey the existing software and tools, starting with well established tools for scientific communication and publication, continuing with computational tools and resources, and finishing with more experimental projects. Established media and tools:

- Text processing and formatting: \LaTeX , Markdown, ...
- Hypertext: HTML and web standards
- Publication/archiving: arXiv, MathSciNet, zbMATH, EuDML
- Web authoring tools: Wordpress, MediaWiki, ...
- Notational standards: MathML, OpenMath, ...
- Notebooks: Mathematica, Jupyter
- Text, reference and metadata-based search: Google Scholar, Semantic Scholar
- Online reference (human oriented): Wikipedia, Mathworld
- Q&A social media: Stack Overflow, Zulip, ...
- Referencing tools and standards: Bibtex, ORCID, ...

Computational tools

- Computer mathematics / symbolic algebra platforms
 - Mathematica, Wolfram Alpha
 - SageMath, CoCalc
 - Specialized systems: GAP, Pari, Singular, Macaulay2, Magma, ...
 - Integrated with a general purpose language: SymPy
 - And more: see Wikipedia's List of computer algebra systems.
- Software development tools
 - Interactive programming languages: R / S, MatLab, Python, Julia, ...
 - Repositories with version control: Git, Github
 - Software libraries: PyPI, MathWorks FileExchange, ...
 - Interactive development environments
- Mathematical databases
 - Online Encyclopedia of Integer Sequences
 - Atlas of Finite Groups, GAP libraries
 - Kreuzer-Skarke database of Calabi-Yau manifolds
 - See <https://mathdb.mathhub.info/> for many more

We will discuss some of these entries in more detail below.

Interactive theorem verification

- Proof assistants:
 - Isabelle/HOL, HOL Light
 - Coq
 - Lean
 - Mizar, Metamath, ...

All of these have libraries of proven theorems = more mathematical databases.

- Automated provers, mostly for first order logic: E, Vampire, ...
- SAT and SMT solvers: Z3, Alt-Ergo, ...
- Flagship verified proofs:
 - The four-color theorem
 - The Feit-Thompson theorem
 - Hales's proof of the Kepler conjecture

We will go into more depth about this topic later.

More experimental or avant-garde projects

- Collaborative textbooks: the Stacks Project
- Search for formulas: SearchOnMath, MathWebSearch
- Web-based math collaborations: Polymath
- Verified mathematics repositories: the QED Manifesto, Mizar, Formal Abstracts
- AI and theorem proving: TacticToe, GamePad, HOList, GPT-F
- Math/CS collaborations: Lean Forward, ...
- Math dataset infrastructure: MDDL
- Computer grading and intelligent tutoring: STACK
- Other mathematical knowledge management projects: GDML, OMDoc, IMKT

Mathematical search

It's not clear that any of these are mature technologies, in the sense that we can only expect incremental improvements. But some of them are widely used. For example, almost everyone here uses search engines and computer typesetting (mostly \LaTeX), most of us read and write papers on arXiv and look at Wikipedia articles, some of us write blogs or Wikipedia articles, and many of us use Mathematica or SageMath, and write programs in Python.

Let's give some examples to illustrate some experimental tools, the problems they solve, and their limitations. We will start with **search**. Many mathematicians, for example Tim Gowers, have emphasized the potential value of a mathematical search engine. What is this?

The idea is, we describe a mathematical concept, and get back a list of documents containing claims, explanations, proofs, algorithms, computer code, *etc.*, all relevant to our query.

Mathematical search

It's not clear that any of these are mature technologies, in the sense that we can only expect incremental improvements. But some of them are widely used. For example, almost everyone here uses search engines and computer typesetting (mostly \LaTeX), most of us read and write papers on arXiv and look at Wikipedia articles, some of us write blogs or Wikipedia articles, and many of us use Mathematica or SageMath, and write programs in Python.

Let's give some examples to illustrate some experimental tools, the problems they solve, and their limitations. We will start with **search**. Many mathematicians, for example Tim Gowers, have emphasized the potential value of a mathematical search engine. What is this?

The idea is, we describe a mathematical concept, and get back a list of documents containing claims, explanations, proofs, algorithms, computer code, *etc.*, all relevant to our query.

The easy case is when we are looking up a theorem or definition with its own name. Thus, if we want to know more about the four square theorem, or Sobolev spaces, we can just type the name into a search engine, and get many relevant responses:

sobolev spaces
🔍

🔍 All
📖 Books
📺 Videos
🖼️ Images
📰 News
⋮ More
⚙️ Settings
🛠️ Tools

About 2,970,000 results (0.49 seconds)

Sobolev space - Wikipedia
https://en.wikipedia.org/wiki/Sobolev_space

In mathematics, a **Sobolev space** is a vector space of functions equipped with a norm that is a combination of L^p -norms of the function itself and its derivatives up to a given order. The derivatives are understood in a suitable weak sense to make the space complete, thus a Banach space.

[Sobolev inequality](#) · [Category:Sobolev spaces](#) · [Sergei Sobolev](#) · [Weak derivative](#)

Hilbert and Sobolev spaces
https://www.ljll.math.upmc.fr/frey/cours/udc/ma691_ch2


by D Hilbert

Hilbert and Sobolev spaces. "The unified character of mathematics lies in its very nature; indeed, mathematics is the foundation of all exact natural sciences."

Sobolev Space – from Wolfram MathWorld
[mathworld.wolfram.com](http://mathworld.wolfram.com/SobolevSpace.html) > ... > [Spaces](#) > [MathWorld Contributors](#) > [Oliveira](#)

by EW Weisstein - 2004

Sobolev Space. For $d \geq 1$, Ω an open subset of \mathbb{R}^d , p in $[1, \infty]$ and s in \mathbb{N} , the **Sobolev space** $W^{s,p}(\mathbb{R}^d)$ is defined by ...



Sobolev space

In mathematics, a Sobolev space is a vector space of functions equipped with a norm that is a combination of L^p -norms of the function itself and its derivatives up to a given order. The derivatives are understood in a suitable weak sense to make the space complete, thus a Banach space. [Wikipedia](#)

[Feedback](#)

🔍
🔍
🔍

While this is very helpful, it is hardly the answer to all of our questions.

Suppose we don't know the name?

Suppose we invent a **new** concept or prove a new theorem? Are we really sure it is new? Maybe somebody already discovered or proved it. So what do we search for?

More than most human pursuits, in mathematics we can state our claims in simple and “universal” ways. Suppose we develop a search engine to look for formulas (including logical expressions) in documents. This would be a start, but of course a concept could be described by many different formulas, require several formulas for its definition, *etc.*. One might wonder how well a textual search could work.

Let's try it out. The web site <https://www.searchonmath.com> offers formula search. Could we find “prime numbers” just from the definition?

While this is very helpful, it is hardly the answer to all of our questions.

Suppose we don't know the name?

Suppose we invent a **new** concept or prove a new theorem? Are we really sure it is new? Maybe somebody already discovered or proved it. So what do we search for?

More than most human pursuits, in mathematics we can state our claims in simple and “universal” ways. Suppose we develop a search engine to look for formulas (including logical expressions) in documents. This would be a start, but of course a concept could be described by many different formulas, require several formulas for its definition, *etc.*. One might wonder how well a textual search could work.

Let's try it out. The web site <https://www.searchonmath.com> offers formula search. Could we find “prime numbers” just from the definition?

While this is very helpful, it is hardly the answer to all of our questions.

Suppose we don't know the name?

Suppose we invent a **new** concept or prove a new theorem? Are we really sure it is new? Maybe somebody already discovered or proved it. So what do we search for?

More than most human pursuits, in mathematics we can state our claims in simple and “universal” ways. Suppose we develop a search engine to look for formulas (including logical expressions) in documents. This would be a start, but of course a concept could be described by many different formulas, require several formulas for its definition, *etc.*. One might wonder how well a textual search could work.

Let's try it out. The web site <https://www.searchonmath.com> offers formula search. Could we find “prime numbers” just from the definition?

While this is very helpful, it is hardly the answer to all of our questions.

Suppose we don't know the name?

Suppose we invent a **new** concept or prove a new theorem? Are we really sure it is new? Maybe somebody already discovered or proved it. So what do we search for?

More than most human pursuits, in mathematics we can state our claims in simple and “universal” ways. Suppose we develop a search engine to look for formulas (including logical expressions) in documents. This would be a start, but of course a concept could be described by many different formulas, require several formulas for its definition, *etc.*. One might wonder how well a textual search could work.

Let's try it out. The web site <https://www.searchonmath.com> offers formula search. Could we find “prime numbers” just from the definition?

While this is very helpful, it is hardly the answer to all of our questions.

Suppose we don't know the name?

Suppose we invent a **new** concept or prove a new theorem? Are we really sure it is new? Maybe somebody already discovered or proved it. So what do we search for?

More than most human pursuits, in mathematics we can state our claims in simple and “universal” ways. Suppose we develop a search engine to look for formulas (including logical expressions) in documents. This would be a start, but of course a concept could be described by many different formulas, require several formulas for its definition, *etc.*. One might wonder how well a textual search could work.

Let's try it out. The web site <https://www.searchonmath.com> offers formula search. Could we find “prime numbers” just from the definition?



$\{ p \in \mathbb{N} : \exists n \in \mathbb{N}, n > 1 \wedge n | p \}$





Add Formula



Limit Search Domain

$$\{ p \in \mathbb{N} : \exists n \in \mathbb{N}, n > 1 \wedge n | p \}$$

I know, this is the set $\{1\}$, but maybe this is close enough.
Type it in into <https://www.searchonmath.com>. The result:

About 8 results in 0.42 seconds.

$P := \{ p \in \mathbb{N} : \forall c \in \mathbb{N},$  <https://math.stackexchange.com/questions/548513> **By definition, how is a prime number represented? ...**
Even numbers can be easily represented as $2n$. Odd numbers as $2n+1$.
 An exactly divisible operation can be defined as ...

$P = \forall n \in \mathbb{N} : \exists a \in S :$  <https://math.stackexchange.com/questions/2051716> **A finite set S satisfying $\forall n \in \mathbb{N}$...**
I'd like some help with this question. Suppose we have some finite set S
 satisfying $\forall n \in \mathbb{N} : \exists a \in S$...

Pretty good! I didn't think this would work. Let's try a harder one.



$\{ f \in L^2(\mathbb{R}) \wedge \frac{df}{dx} \in L^2(\mathbb{R}) \}$



Add Formula

Limit Search Domain

$$\{f \in L^2(\mathbb{R}) \wedge \frac{df}{dx} \in L^2(\mathbb{R})\}$$

About 2 results in 0.33 seconds.

$$\frac{\partial^{(k)}}{\partial x^k} f, \frac{\partial^{(k)}}{\partial y^k} f \in L^2(\mathbb{R}) \quad \forall k \in \mathbb{N}$$



[Situation of Nirenberg Sobolev embedding Ques...](https://math.stackexchange.com/questions/680404)

<https://math.stackexchange.com/questions/680404>

Suppose $f \in L^2(\mathbb{R}^2)$ with compact support and $\frac{\partial^{(k)}}{\partial x^k} f, \frac{\partial^{(k)}}{\partial y^k} f \in L^2(\mathbb{R}^2)$



$$\forall x \in \mathcal{O} \quad \partial_{x_i} k(x, \cdot) \in L^1(\mathbb{R}^n)$$



[\\$\partial_{x_i} k\(x, \cdot\) \in L^1\(\mathbb{R}^n\)\\$](https://math.stackexchange.com/questions/1695718)

<https://math.stackexchange.com/questions/1695718>

Let $\mathcal{O} \subset \mathbb{R}^n$ be open, with $\mathcal{O} \subset \mathbb{R}^n$ open, be such that $\forall x \in \mathcal{O}$



Not as good, but at least we got names we could try in another search. And actually our previous good result was a bit of a cheat as it used the fact that the symbol p is often used to denote a prime. Consider

search
 \int_n Math

$\{ q \in \mathbb{N} : \exists n \in \mathbb{N} : n|q \wedge n > 1 \wedge n < q \}$



Add Formula

Limit Search Domain

$$\{ q \in \mathbb{N} : \exists n \in \mathbb{N} : n|q \wedge n > 1 \wedge n < q \}$$

About 9 results in 0.18 seconds.

$$\forall n \in \mathbb{N} : \exists n \in \mathbb{N} : (n > 1)$$

Prove that \mathbb{Q} is complete by showing that \mathbb{Q} ...

<https://math.stackexchange.com/questions/2462750>

Question: How do I show that \mathbb{Q} , which is shorthand for $\mathbb{Q}_0(\mathbb{N})$, is closed in \mathbb{R} ? I've tried to wo...

$$\forall u \subseteq \mathbb{N} : \forall n \in \mathbb{N} : (|u| = n)$$

Formalizing sentences in propositional logic Ans...

<https://math.stackexchange.com/questions/2273219>

I'm studying for an introductory mathematical logic exam. Could you help me with formalizing the following conditions in...



A better search engine would have to “anonymize” the variables and try out many substitutions. See Kohlhasse *et al*'s work on `MathWebSearch` for some of the issues. But there are many limitations of any formula-based search:

- Many concepts cannot be described by a simple formula in terms of standard concepts. One must build them up using two or three formulas, or else use a complicated non-canonical formula.
- Even when one has a simple definition, to refer to the standard concepts, one has to know and use the right formulations for them. Even when concepts are standard, their formulations are often not.

An example:

- $\text{normal}(H, G) := H \in \text{subgroups}(G) \wedge \forall g \in G, gHg^{-1} \cong H$
- $\forall H \in \text{subgroups}(G) : \text{normal}(H, G) \Rightarrow H \cong \text{trivial} \vee H \cong G$

Do we have to remember “normal”, or $H \triangleleft G$, or ?

Perhaps a good mathematical search engine will need to be based on a systematic library of mathematical definitions. Later in the talk we will discuss a proposal to make one, the Formal Abstracts project.

But let's suppose this works and we find relevant results – formulas, claims, explanations, proofs, code. What do we do next?

- 1 At one extreme, we study the results, understand them, and rederive everything we actually use.
- 2 At the other, we trust the results and copy them into our own work.

Although (1) might seem virtuous and (2) lazy, of course this is simplistic. We all use many well established and well understood results every day, and it can take a long time to rederive or even check all the details. Life is too short to always do this.

Perhaps a good mathematical search engine will need to be based on a systematic library of mathematical definitions. Later in the talk we will discuss a proposal to make one, the Formal Abstracts project.

But let's suppose this works and we find relevant results – formulas, claims, explanations, proofs, code. What do we do next?

- 1 At one extreme, we study the results, understand them, and rederive everything we actually use.
- 2 At the other, we trust the results and copy them into our own work.

Although (1) might seem virtuous and (2) lazy, of course this is simplistic. We all use many well established and well understood results every day, and it can take a long time to rederive or even check all the details. Life is too short to always do this.

But, even if we want to use well established and well understood search results (claims, proofs, code), how do we know they are correct? This is at least as serious a difficulty with building on previous work as finding it (perhaps more).

Can we systematize this? Some approaches:

- Curation – only use search results published in journals or in curated software libraries.
- Validation – only use results which we can check. For example, a subroutine which implements a function $f(x)$, can be spot-checked against known pairs $(x, f(x))$.
- Certification – only use results which come with a certificate of correctness. For example, a theorem along with its proof.

Later we will discuss the state of the art in automated theorem verification, with this motivation in mind. But first let's talk about other areas where we can expect progress.

But, even if we want to use well established and well understood search results (claims, proofs, code), how do we know they are correct? This is at least as serious a difficulty with building on previous work as finding it (perhaps more).

Can we systematize this? Some approaches:

- Curation – only use search results published in journals or in curated software libraries.
- Validation – only use results which we can check. For example, a subroutine which implements a function $f(x)$, can be spot-checked against known pairs $(x, f(x))$.
- Certification – only use results which come with a certificate of correctness. For example, a theorem along with its proof.

Later we will discuss the state of the art in automated theorem verification, with this motivation in mind. But first let's talk about other areas where we can expect progress.

But, even if we want to use well established and well understood search results (claims, proofs, code), how do we know they are correct? This is at least as serious a difficulty with building on previous work as finding it (perhaps more).

Can we systematize this? Some approaches:

- Curation – only use search results published in journals or in curated software libraries.
- Validation – only use results which we can check. For example, a subroutine which implements a function $f(x)$, can be spot-checked against known pairs $(x, f(x))$.
- Certification – only use results which come with a certificate of correctness. For example, a theorem along with its proof.

Later we will discuss the state of the art in automated theorem verification, with this motivation in mind. But first let's talk about other areas where we can expect progress.

Computational mathematics

Many of the great mathematicians – Euler, Gauss and Ramanujan come to mind – were renowned for the calculational abilities, which were the basis for many mathematical discoveries.

This tradition continues and has been enhanced by the use of computers. Leaving aside computer-aided proofs of earlier conjectures, a few discoveries which were made this way include

- Many properties of chaos in dynamical systems.
- Most of the original constructions of the sporadic finite groups.
- The Birch Swinnerton-Dyer conjecture, based on calculations done on the EDSAC-2 computer at Cambridge.

Numerical experiment is now a central part of number theory, see for example the web site of the Simons Collaboration on Arithmetic Geometry, Number Theory, and Computation.

Still, continuing any specific program eventually runs into exponentially large computing problems, because of the curse of dimensionality, the existence of NP hard problems, *etc.* .

So what are **new** ideas and tools which will lead to progress by 2030 ?

- Faster computers, more storage, better infrastructure
- Satisfiability modulo theories, SAT+CAS, ...
- Neural networks – the subject of many talks at this workshop
- Advances in statistics

The first of these, while not involving deep concepts, can still make a big difference to research. As an example from physics, there was a large effort starting in the 80's to do numerical simulations of lattice gauge theory, to compute masses of hadrons *ab initio*. This problem was essentially solved in the 2000's, mostly thanks to cheap supercomputers (clusters with GPUs).

If we were within a factor of 1000 (time, space, ...) of solving a problem in 2019, it will probably be solved using the same methods by 2030.

SAT, SMT and SAT+CAS

SAT solvers: solve problems in propositional logic, given as a list of clauses.

Math example: the Boolean Pythagorean triple problem, split $\{1 \dots N\}$ into two subsets such that neither subset contains a triple $a^2 + b^2 = c^2$. Possible for $N = 7824$ and not for $N = 7825$, as shown by Heule, Kullmann, and Marek in 2016 (producing “the world’s longest proof”).

In this problem the Boolean variables are the assignments of each number to a subset, and the logical clauses are easy to derive. Many problems involve more algebra, or reasoning in other domains. An SMT (Satisfiability Modulo Theories) solver combines some other decision algorithm with a SAT solver. Often the SAT solver can reach its conclusions without evaluating many of the constraints, so this will be much faster. The “SAT+CAS” variant combines a SAT solver with a general computer algebra system, which checks proposed solutions and can return “conflict clauses” to the SAT solver.

SAT, SMT and SAT+CAS

SAT solvers: solve problems in propositional logic, given as a list of clauses.

Math example: the Boolean Pythagorean triple problem, split $\{1 \dots N\}$ into two subsets such that neither subset contains a triple $a^2 + b^2 = c^2$. Possible for $N = 7824$ and not for $N = 7825$, as shown by Heule, Kullmann, and Marek in 2016 (producing “the world’s longest proof”).

In this problem the Boolean variables are the assignments of each number to a subset, and the logical clauses are easy to derive. Many problems involve more algebra, or reasoning in other domains. An SMT (Satisfiability Modulo Theories) solver combines some other decision algorithm with a SAT solver. Often the SAT solver can reach its conclusions without evaluating many of the constraints, so this will be much faster. The “SAT+CAS” variant combines a SAT solver with a general computer algebra system, which checks proposed solutions and can return “conflict clauses” to the SAT solver.

Problems suitable for this approach tend to be those in which the search space has a natural Boolean encoding. There are many examples in <https://arxiv.org/abs/1907.04408> (the MathCheck project):

- The Williamson conjecture: find four symmetric $n \times n$ matrices with ± 1 entries such that

$$A^2 + B^2 + C^2 + D^2 = 4n \cdot \text{id}. \quad (1)$$

These can be arranged into a $4n \times 4n$ Hadamard matrix satisfying $HH^t = 4n \cdot \text{id}$. Strangely enough, these exist for all $n < 35$, and all even $n \leq 70$, but not for $n = 35$.

- Golay pairs: polynomials f, g with coefficients from $\{1, i, -1, -i\}$ such that $|f(z)|^2 + |g(z)|^2$ is constant on the unit circle.
- 3×3 matrix multiplication using fewer than 27 scalar multiplications. One can find many ways using 23 and so far, none using 22.

Neural networks

Neural networks and deep learning are useful not only for neuroscience and AI, but can be applied in many numerical computations.

Function fitting (interpolation, extrapolation) is ubiquitous. A few examples:

- Optimization: model the objective function.
- PDE's: interpolate solutions at subgrid scales.
- quantum chemistry: predict ground state energies of molecules.

Another great advantage of NN's is that one can save a lot of time writing programs, instead training the NN on a large dataset – this is particularly useful for exploratory work.

The theory of what function classes can be approximated or what tasks learned is in its infancy – raising very interesting mathematical questions.

Neural networks

Neural networks and deep learning are useful not only for neuroscience and AI, but can be applied in many numerical computations.

Function fitting (interpolation, extrapolation) is ubiquitous. A few examples:

- Optimization: model the objective function.
- PDE's: interpolate solutions at subgrid scales.
- quantum chemistry: predict ground state energies of molecules.

Another great advantage of NN's is that one can save a lot of time writing programs, instead training the NN on a large dataset – this is particularly useful for exploratory work.

The theory of what function classes can be approximated or what tasks learned is in its infancy – raising very interesting mathematical questions.

Advances in statistics

Statistics is hardly new, but one could argue that with less need for computational efficiency, one is more free to use general approaches based on simple concepts:

- Generative models: give entire probability distribution of data.
- Bayesian statistics: turn around model \Rightarrow data to infer $P(\text{model})$.
- Information theory: KL divergence, variational methods
- Distances between observations or measures: Wasserstein distance and optimal transport, ...

Given two measures μ and ν on a metric space M , the p 'th Wasserstein distance between them is

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu, \nu)} \int_{M \times M} d(x, y)^p d\gamma(x, y) \right)^{1/p} \quad (2)$$

where $\int_{M_1} d\gamma = \mu$, resp. M_2 and ν .

An interesting trend in physics and other hard sciences is to replace hand-crafted concepts and tools for data analysis, with more general statistical tools. An example from particle physics is Thaler *et al*, <http://arxiv.org/abs/1902.02346>, “The Metric Space of Collider Events.” They use Wasserstein W_1 as a distance between collider events, considered as energy distributions μ, ν .



What might a math textbook look like in 2030?

Let's have an example in mind: an introductory textbook on ordinary differential equations for math majors. The primary goal is clear: to help students who understand the prerequisites to master a specified body of knowledge, usually at the breadth and depth that can be covered in a semester or year-long university course. It has chapters with a linear or tree-like dependency structure, clear explanations, consistent notation, and numerous exercises of various levels of difficulty (and with the difficulty clearly marked). Ideally it has an index pointing to the definition and each significant reference for every concept and notation, and references for further reading.

We could go into the design choices in more detail, but the main point is, book technology requires them to be fixed. In doing this, one generally assumes a particular use (teaching a university course). Computer technology is far more flexible. On the other hand, some design choices contribute to effective learning. How to proceed?

What might a math textbook look like in 2030?

Let's have an example in mind: an introductory textbook on ordinary differential equations for math majors. The primary goal is clear: to help students who understand the prerequisites to master a specified body of knowledge, usually at the breadth and depth that can be covered in a semester or year-long university course. It has chapters with a linear or tree-like dependency structure, clear explanations, consistent notation, and numerous exercises of various levels of difficulty (and with the difficulty clearly marked). Ideally it has an index pointing to the definition and each significant reference for every concept and notation, and references for further reading.

We could go into the design choices in more detail, but the main point is, book technology requires them to be fixed. In doing this, one generally assumes a particular use (teaching a university course). Computer technology is far more flexible. On the other hand, some design choices contribute to effective learning. How to proceed?

What might a math textbook look like in 2030?

Let's have an example in mind: an introductory textbook on ordinary differential equations for math majors. The primary goal is clear: to help students who understand the prerequisites to master a specified body of knowledge, usually at the breadth and depth that can be covered in a semester or year-long university course. It has chapters with a linear or tree-like dependency structure, clear explanations, consistent notation, and numerous exercises of various levels of difficulty (and with the difficulty clearly marked). Ideally it has an index pointing to the definition and each significant reference for every concept and notation, and references for further reading.

We could go into the design choices in more detail, but the main point is, book technology requires them to be fixed. In doing this, one generally assumes a particular use (teaching a university course). Computer technology is far more flexible. On the other hand, some design choices contribute to effective learning. How to proceed?

Let's develop a wish list, ranging from more modest ideas to more ambitious ones, and then ask how feasible each wish might be.

First, some existing capabilities could be added and/or improved:

- Adaptations for modern media. All references within and without a document should be hyperlinks, some should be pop-up links. It should be easy for a student to highlight passages and record his or her own notes and connections.
- Automated grading of exercises, ideally accepting sketches of work and giving partial credit (see Chris Sangwin's talk at www.icms.org.uk/downloads/bigproof/Sangwin.pdf).
- Social learning – exchange of hints and explanations through Q&A sites, finding online collaborators to discuss problems.
- More adaptations. Computational notebooks allow expanding/contracting passages. A text could have a flow chart, not just of chapters but of concepts. Ideally the flow chart would be generated (semi-)automatically. The reader could expand only the nodes of interest.

Let's develop a wish list, ranging from more modest ideas to more ambitious ones, and then ask how feasible each wish might be.

First, some existing capabilities could be added and/or improved:

- Adaptations for modern media. All references within and without a document should be hyperlinks, some should be pop-up links. It should be easy for a student to highlight passages and record his or her own notes and connections.
- Automated grading of exercises, ideally accepting sketches of work and giving partial credit (see Chris Sangwin's talk at www.icms.org.uk/downloads/bigproof/Sangwin.pdf).
- Social learning – exchange of hints and explanations through Q&A sites, finding online collaborators to discuss problems.
- More adaptations. Computational notebooks allow expanding/contracting passages. A text could have a flow chart, not just of chapters but of concepts. Ideally the flow chart would be generated (semi-)automatically. The reader could expand only the nodes of interest.

Let's develop a wish list, ranging from more modest ideas to more ambitious ones, and then ask how feasible each wish might be.

First, some existing capabilities could be added and/or improved:

- Adaptations for modern media. All references within and without a document should be hyperlinks, some should be pop-up links. It should be easy for a student to highlight passages and record his or her own notes and connections.
- Automated grading of exercises, ideally accepting sketches of work and giving partial credit (see Chris Sangwin's talk at www.icms.org.uk/downloads/bigproof/Sangwin.pdf).
- Social learning – exchange of hints and explanations through Q&A sites, finding online collaborators to discuss problems.
- More adaptations. Computational notebooks allow expanding/contracting passages. A text could have a flow chart, not just of chapters but of concepts. Ideally the flow chart would be generated (semi-)automatically. The reader could expand only the nodes of interest.

Let's develop a wish list, ranging from more modest ideas to more ambitious ones, and then ask how feasible each wish might be.

First, some existing capabilities could be added and/or improved:

- Adaptations for modern media. All references within and without a document should be hyperlinks, some should be pop-up links. It should be easy for a student to highlight passages and record his or her own notes and connections.
- Automated grading of exercises, ideally accepting sketches of work and giving partial credit (see Chris Sangwin's talk at www.icms.org.uk/downloads/bigproof/Sangwin.pdf).
- Social learning – exchange of hints and explanations through Q&A sites, finding online collaborators to discuss problems.
- More adaptations. Computational notebooks allow expanding/contracting passages. A text could have a flow chart, not just of chapters but of concepts. Ideally the flow chart would be generated (semi-)automatically. The reader could expand only the nodes of interest.

Let's develop a wish list, ranging from more modest ideas to more ambitious ones, and then ask how feasible each wish might be.

First, some existing capabilities could be added and/or improved:

- Adaptations for modern media. All references within and without a document should be hyperlinks, some should be pop-up links. It should be easy for a student to highlight passages and record his or her own notes and connections.
- Automated grading of exercises, ideally accepting sketches of work and giving partial credit (see Chris Sangwin's talk at www.icms.org.uk/downloads/bigproof/Sangwin.pdf).
- Social learning – exchange of hints and explanations through Q&A sites, finding online collaborators to discuss problems.
- More adaptations. Computational notebooks allow expanding/contracting passages. A text could have a flow chart, not just of chapters but of concepts. Ideally the flow chart would be generated (semi-)automatically. The reader could expand only the nodes of interest.

Going beyond this, we would like our students to learn concepts, and also techniques and tools for solving their problems. Both take time and often a course in computational methods comes after an introductory course.

Here is a textbook which takes a different approach:



This is a textbook on advanced classical mechanics, including Lagrangian and Hamiltonian mechanics, canonical transformations and perturbation theory, and chaos theory. The authors Gerald Jay Sussman and Jack Wisdom are renowned experts in computer science/AI and planetary science/celestial mechanics respectively.

While its table of contents is similar to others on the topic, the book integrates computation from the start, providing a platform based on Scheme which facilitates not only numerical experiments, but also symbolic computations. Basic principles such as the meaning of a Lagrangian functional and the derivation of equations of motion are not just explained, but presented with computational examples. For example, in chapter 1, all of the steps starting from the initial Lagrangian or equations of motion for a driven pendulum, to the numerical integrator and results, are given with explicit (and short!) computer code.

This is a textbook on advanced classical mechanics, including Lagrangian and Hamiltonian mechanics, canonical transformations and perturbation theory, and chaos theory. The authors Gerald Jay Sussman and Jack Wisdom are renowned experts in computer science/AI and planetary science/celestial mechanics respectively.

While its table of contents is similar to others on the topic, the book integrates computation from the start, providing a platform based on Scheme which facilitates not only numerical experiments, but also symbolic computations. Basic principles such as the meaning of a Lagrangian functional and the derivation of equations of motion are not just explained, but presented with computational examples. For example, in chapter 1, all of the steps starting from the initial Lagrangian or equations of motion for a driven pendulum, to the numerical integrator and results, are given with explicit (and short!) computer code.

Interactive theorem verification

In the 50's and 60's, the pioneers of AI developed automatic theorem provers, which generate logical deductions and search for proofs of given logical statements. Concurrently, the subfield of formal methods was developed, in which computer programs were given precise semantics allowing them to be rigorously verified.

This is of great practical value, especially for programs (an airplane autopilot, a CPU floating point unit) where mistakes can be extremely expensive. Thus it has been pursued intensively for decades, the formal methods community is fairly large and well-funded, and most of the systems we cited (Isabelle and Coq, though not Mizar) have software verification as the primary application.

Interactive theorem verification

In the 50's and 60's, the pioneers of AI developed automatic theorem provers, which generate logical deductions and search for proofs of given logical statements. Concurrently, the subfield of formal methods was developed, in which computer programs were given precise semantics allowing them to be rigorously verified.

This is of great practical value, especially for programs (an airplane autopilot, a CPU floating point unit) where mistakes can be extremely expensive. Thus it has been pursued intensively for decades, the formal methods community is fairly large and well-funded, and most of the systems we cited (Isabelle and Coq, though not Mizar) have software verification as the primary application.

As a prototypical example of software verification, let us briefly discuss sorting a list into alphabetical order. In an algorithms course one learns that a list of N elements can be sorted in worst case time $N \log N$, but the algorithms (quicksort, heapsort, ...) are a bit tricky. On the other hand, logically defining the problem of sorting is not difficult. In Coq we can say

```

Definition is_a_sorting_algorithm (f: list nat → list nat) :=
  ∀ al, Permutation al (f al) ∧ sorted (f al).

Definition sorted' (al: list nat) :=
  ∀ i j, i < j < length al → nth i al 0 ≤ nth j al 0.

Inductive Permutation : list A -> list A -> Prop :=
| perm_nil: Permutation [] []
| perm_skip x l l' : Permutation l l' -> Permutation (x::l) (x::l')
| perm_swap x y l : Permutation (y::x::l) (x::y::l)
| perm_trans l l' l'' :
  Permutation l l' -> Permutation l' l'' -> Permutation l l''.

```

In <https://softwarefoundations.cis.upenn.edu/vfa-current/Sort.html> one can see formally verified proofs that runnable sorting programs satisfy this specification.

The Fundamental Theorem of Algebra

Let's look a bit at a mathematical example, the fundamental theorem of algebra. As we all know, this states that the field \mathbb{C} of complex numbers is algebraically closed, in other words every nonconstant polynomial $f(z)$ has a root.

This claim can be easily formalized: in the Lean theorem proving language, we can say

$$\begin{aligned} \text{lemma exists_root} \quad & \{f : \text{polynomial } \mathbb{C}\} \quad (\text{hf} : 0 < \text{degree } f) : \\ & \exists z : \mathbb{C}, f.\text{eval } z = 0 := \end{aligned} \quad (3)$$

followed by the proof.

Here is an informal proof. We start by assuming that f has no zero, to get a contradiction. (A constructive proof exists but is longer.)

- 1 We first show that $|f(z)|$ attains its minimum at some point z_0 . A polynomial goes to infinity at infinity, so the infimum of f is contained in a closed bounded region R . Since $|f|$ is continuous, the image of R is closed and bounded, so it contains its infimum.
- 2 Expand around the location z_0 of the minimum by writing

$$f(z) = f(z_0) + (z - z_0)^n g(z)$$

for some polynomial $g(z)$ such that $g(z_0) \neq 0$.

- 3 Now, consider a small circle $z = z_0 + \delta e^{i\theta}$. If we neglect the variation of $g(z)$ and look at $F(z) = f(z_0) + (z - z_0)^n g(z_0)$, it is easy to show that z_0 cannot be a minimum of $|F(z)|$, since $(z - z_0)^n$ takes every possible phase.
- 4 Intuitively, we then want to choose δ small enough such that we can neglect the variation of $g(z)$, so z_0 cannot be a minimum of $|f(z)|$, a contradiction. After a bit of algebra, it turns out that $|g(z) - g(z_0)| < |g(z_0)| \forall |z - z_0| \leq \delta$ suffices.

The Lean version of this proof takes about 100 lines: here is part 1.

```

12 lemma exists_forall_abs_polynomial_eval_le (p : polynomial ℂ) :
13   ∃ x, ∀ y, (p.eval x).abs ≤ (p.eval y).abs :=
14   if hp0 : 0 < degree p
15   then let (r, hr0, hr) := polynomial.tendsto_infinity complex.abs hp0 ((p.eval 0).abs) in
16     let (x, hx1, hx2) := exists_forall_le_of_compact_of_continuous (λ y, (p.eval y).abs)
17       (continuous_abs.comp p.continuous_eval)
18       (closed_ball 0 r) (proper_space.compact_ball _ _)
19       (set.ne_empty_iff_exists_mem.2 (0, by simp [le_of_lt hr0])) in
20     (x, λ y, if hy : y.abs ≤ r then hx2 y $ by simp [complex.dist_eq] using hy
21       else le_trans (hx2 _ (by simp [le_of_lt hr0])) (le_of_lt (hr y (lt_of_not_ge hy))))
22   else {p.coeff 0, by rw [eq_C_of_degree_le_zero (le_of_not_gt hp0)]; simp}
--

```

- The statement being proved is on line 13 - it is clear and intuitive.
- The language is “computerese” - but this is a question of taste and one can display the same content in more math-friendly notations.
- Unlike the informal proof, we had to give many propositions their own names and calling conventions, which also hurts readability.
- Commands like `rw` and `simp` are “tactics,” explicit instructions to the proof verifier. These are procedural and tricky to get right.

Today's ITV systems incorporate many advances in logic, such as dependent type theory (in Coq and Lean), and many advances in computer science. But despite all this work, theorem verification is more akin to programming than to any of the traditional skills of a mathematical scientist. And the many differences with informal proofs which we just cited, while each fairly simple, add up. At present ITV is hard to learn and use.

Still, many people believe that formalization and verification is a central part of the relation between computers and mathematics. This even includes theoretical physicists and others for whom rigorous proof is not a primary goal. It is hard to get a computer to understand anything, and here is a way for it to “understand truth.” So how to use this?

- Definitions are easier to write than proofs: focus on these?
- Perhaps ITVs need more reasoning methods than deductive logic? Say counterexamples, heuristics, *etc.*. Omitted in this talk.
- Will machine learning and AI help?

Today's ITV systems incorporate many advances in logic, such as dependent type theory (in Coq and Lean), and many advances in computer science. But despite all this work, theorem verification is more akin to programming than to any of the traditional skills of a mathematical scientist. And the many differences with informal proofs which we just cited, while each fairly simple, add up. At present ITV is hard to learn and use.

Still, many people believe that formalization and verification is a central part of the relation between computers and mathematics. This even includes theoretical physicists and others for whom rigorous proof is not a primary goal. It is hard to get a computer to understand anything, and here is a way for it to “understand truth.” So how to use this?

- Definitions are easier to write than proofs: focus on these?
- Perhaps ITVs need more reasoning methods than deductive logic? Say counterexamples, heuristics, *etc.*. Omitted in this talk.
- Will machine learning and AI help?

Today's ITV systems incorporate many advances in logic, such as dependent type theory (in Coq and Lean), and many advances in computer science. But despite all this work, theorem verification is more akin to programming than to any of the traditional skills of a mathematical scientist. And the many differences with informal proofs which we just cited, while each fairly simple, add up. At present ITV is hard to learn and use.

Still, many people believe that formalization and verification is a central part of the relation between computers and mathematics. This even includes theoretical physicists and others for whom rigorous proof is not a primary goal. It is hard to get a computer to understand anything, and here is a way for it to “understand truth.” So how to use this?

- Definitions are easier to write than proofs: focus on these?
- Perhaps ITVs need more reasoning methods than deductive logic? Say counterexamples, heuristics, *etc.*. Omitted in this talk.
- Will machine learning and AI help?

Formal Abstracts

Tom Hales at U Pittsburgh has begun a project to create an online repository of formal abstracts, meaning statements of the main results of a mathematical paper, expressed in formal terms. Proofs would not be required, but it should be possible in principle to prove every abstract true or false. This project has many parts – here are a few (based on discussions with Tom):

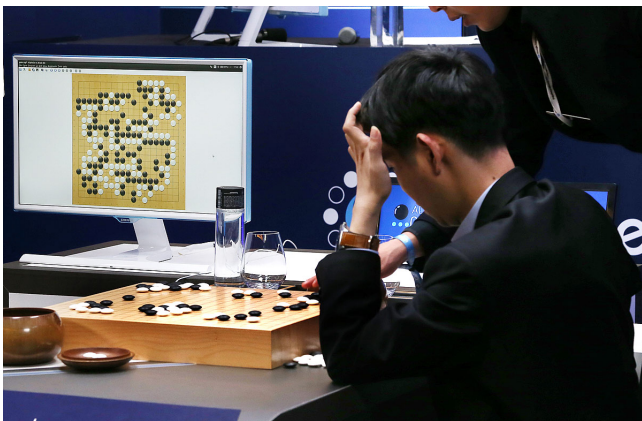
- A solid ITV with dependent types – Lean.
- A library of standard concepts which can be used by abstracts, probably covering all of advanced undergraduate/early graduate level mathematics. A very rough estimate of the size is about 50,000 definitions filling 10,000 pages.
- Abstracts can be written in a controlled natural language which looks like standard mathematical text.
- Interactive tools to help search for, read and write abstracts.

While ambitious, such a system could be fully operational with its library in less than five years.

Formalization of research level mathematics

Here are some recent and current projects in formalization of research mathematics, including central results with substantial proofs, and frontier topics. See recent articles and blogs of Kevin Buzzard for the latest developments.

- Schemes in algebraic geometry (Buzzard *et al* arXiv:2101.02602).
- Peter Scholze’s “Liquid Tensor Experiment” – formalize the proof of a key theorem about condensed abelian groups in *Lectures on Analytic Geometry* by Clausen and Scholze. Completed on May 28 by Johan Commelin and the Lean community.
- Perfectoid spaces (Buzzard, Commelin, Massot).
- Independence of the continuum hypothesis (Han and van Doorn, arXiv:2102.02901).
- In progress (Massot *et al*): the h-principle for open and ample first order differential relations, and its application to sphere eversion (a homotopy of embeddings $S^2 \rightarrow \mathbb{R}^3$ which exchanges inside and outside).



As for the third question, clearly AI has made transformative progress over the last decade. Let's come back to this after discussing some more applications.

Big proof

CAR has attracted attention from mathematicians for its potential to solve some difficult challenges. One of the recognized challenges is “big proof,” the ability to work with, verify and communicate mathematical proofs which are too large for any individual to fully comprehend. The classic examples are proofs which involve case-by-case analysis of a vast number of cases, such as the proofs of the four-color theorem and of the Kepler conjecture.

In the proof of the four-color theorem (Appel and Haken 1976, Robertson *et al* 1995), one shows that every planar graph must contain as a subgraph, one out of a list of “reducible” configurations from which one can remove an edge while maintaining four-colorability. Thus, by induction, all graphs are four-colorable. And it turns out that such a list can consist of subgraphs with non-negative discrete curvature (arity) and a bound on the perimeter length, so they can be enumerated.

Big proof

CAR has attracted attention from mathematicians for its potential to solve some difficult challenges. One of the recognized challenges is “big proof,” the ability to work with, verify and communicate mathematical proofs which are too large for any individual to fully comprehend. The classic examples are proofs which involve case-by-case analysis of a vast number of cases, such as the proofs of the four-color theorem and of the Kepler conjecture.

In the proof of the four-color theorem (Appel and Haken 1976, Robertson *et al* 1995), one shows that every planar graph must contain as a subgraph, one out of a list of “reducible” configurations from which one can remove an edge while maintaining four-colorability. Thus, by induction, all graphs are four-colorable. And it turns out that such a list can consist of subgraphs with non-negative discrete curvature (arity) and a bound on the perimeter length, so they can be enumerated.

This is a fine approach to proving the four-color theorem, except that the smallest known list has 633 subgraphs, and the computations required to prove its sufficiency are lengthy, even for a computer. (This list is not unique and perhaps the set of possible lists could be better understood.)

In 1611, Kepler conjectured that the most efficient (densest) way to pack spheres in three dimensions is the hexagonal close packing (and an infinite set of variations on it). This was proven in 1998 by Thomas Hales, with computer assistance. The proof involves solving over 100,000 linear programming problems.

Both of these proofs have been formally verified, meaning not just that the computations have been checked, but that the arguments by which these results imply the mathematical theorem have been formalized and verified. For the four-color theorem this was done in 2005 by Gonthier and collaborators (see Gonthier 2008 in the Notices of the AMS). For Kepler this was finished in 2014 by Hales and collaborators (the Flyspeck project).

This is a fine approach to proving the four-color theorem, except that the smallest known list has 633 subgraphs, and the computations required to prove its sufficiency are lengthy, even for a computer. (This list is not unique and perhaps the set of possible lists could be better understood.)

In 1611, Kepler conjectured that the most efficient (densest) way to pack spheres in three dimensions is the hexagonal close packing (and an infinite set of variations on it). This was proven in 1998 by Thomas Hales, with computer assistance. The proof involves solving over 100,000 linear programming problems.

Both of these proofs have been formally verified, meaning not just that the computations have been checked, but that the arguments by which these results imply the mathematical theorem have been formalized and verified. For the four-color theorem this was done in 2005 by Gonthier and collaborators (see Gonthier 2008 in the Notices of the AMS). For Kepler this was finished in 2014 by Hales and collaborators (the Flyspeck project).

This is a fine approach to proving the four-color theorem, except that the smallest known list has 633 subgraphs, and the computations required to prove its sufficiency are lengthy, even for a computer. (This list is not unique and perhaps the set of possible lists could be better understood.)

In 1611, Kepler conjectured that the most efficient (densest) way to pack spheres in three dimensions is the hexagonal close packing (and an infinite set of variations on it). This was proven in 1998 by Thomas Hales, with computer assistance. The proof involves solving over 100,000 linear programming problems.

Both of these proofs have been formally verified, meaning not just that the computations have been checked, but that the arguments by which these results imply the mathematical theorem have been formalized and verified. For the four-color theorem this was done in 2005 by Gonthier and collaborators (see Gonthier 2008 in the Notices of the AMS). For Kepler this was finished in 2014 by Hales and collaborators (the Flyspeck project).

There are also important theorems for which the proof, while composed of subarguments which can each be understood and checked by a human, in their totality strain the limits of human collaboration.

Perhaps the outstanding example is the classification of finite simple groups. This is a very central result in mathematics, whose full statement (including the definitions of the groups) can be made in perhaps 40 pages. According to Solomon 2018, the proof is spread over hundreds of articles, some of which depend on unpublished work. There is an ongoing project to publish a complete proof by 2023, consisting of 12 volumes, each many hundreds of pages long.

In physics and the exact sciences, although proofs are valuable, the central focus is on calculations whose results can be compared with experiment. Such calculations can be vast by any standard: millions of Feynman diagrams, thousands of particles or atoms, *etc.* And despite the efforts of multiple collaborations and scrutiny of reviewers, sometimes mistakes take a while to catch.

There are also important theorems for which the proof, while composed of subarguments which can each be understood and checked by a human, in their totality strain the limits of human collaboration.

Perhaps the outstanding example is the classification of finite simple groups. This is a very central result in mathematics, whose full statement (including the definitions of the groups) can be made in perhaps 40 pages. According to Solomon 2018, the proof is spread over hundreds of articles, some of which depend on unpublished work. There is an ongoing project to publish a complete proof by 2023, consisting of 12 volumes, each many hundreds of pages long.

In physics and the exact sciences, although proofs are valuable, the central focus is on calculations whose results can be compared with experiment. Such calculations can be vast by any standard: millions of Feynman diagrams, thousands of particles or atoms, *etc.* And despite the efforts of multiple collaborations and scrutiny of reviewers, sometimes mistakes take a while to catch.

$g - 2$ of the muon

An example which is somewhat infamous in particle physics is the calculation of $g - 2$ for the muon, in other words its magnetic moment. A muon has a lifetime of about 2 microseconds, long enough to do precision measurements, but not to store. Thus the muons must be created in a particle accelerator and this makes the measurement challenging. Nevertheless there has been a major effort to do it and $g - 2$ has been measured to better than a part in a billion.

The great interest in $g - 2$ is because it can be calculated to high precision in the Standard Model. Its value depends not just on the known particles, but on hypothetical new particles which present-day accelerators cannot produce directly. At present the theoretical and experimental numbers actually differ by more than 3 standard deviations, and a new experiment has begun at Fermilab to improve the precision to 5 standard deviations. If the difference is real, there must be new physics beyond the Standard Model.

$g - 2$ of the muon

An example which is somewhat infamous in particle physics is the calculation of $g - 2$ for the muon, in other words its magnetic moment. A muon has a lifetime of about 2 microseconds, long enough to do precision measurements, but not to store. Thus the muons must be created in a particle accelerator and this makes the measurement challenging. Nevertheless there has been a major effort to do it and $g - 2$ has been measured to better than a part in a billion.

The great interest in $g - 2$ is because it can be calculated to high precision in the Standard Model. Its value depends not just on the known particles, but on hypothetical new particles which present-day accelerators cannot produce directly. At present the theoretical and experimental numbers actually differ by more than 3 standard deviations, and a new experiment has begun at Fermilab to improve the precision to 5 standard deviations. If the difference is real, there must be new physics beyond the Standard Model.

Clearly comparing experiment and theory in the ninth decimal place is a tricky business. And this is not the first time $g - 2$ has disagreed between theory and experiment. For several years (1996-2001) there was also a 3 sigma disagreement, inspiring many speculations about new physics.

This disagreement turned out to be due to a theoretical mistake. The $g - 2$ calculation is particularly tricky, not so much because it involves thousands of Feynman diagrams (this was systematized long ago) but because it requires combining effects from several different sources: QED, the weak interactions, and effects of virtual hadrons, in particular an effect called “hadronic light by light scattering.” This last effect cannot be measured, nor can it be calculated from first principles; it is calculated using phenomenological models of hadrons.

Being something of a weak link, this part of the calculation was scrutinized particularly carefully, with several groups each using their own preferred models. Thus it was a surprise when a mistake was discovered.

Clearly comparing experiment and theory in the ninth decimal place is a tricky business. And this is not the first time $g - 2$ has disagreed between theory and experiment. For several years (1996-2001) there was also a 3 sigma disagreement, inspiring many speculations about new physics.

This disagreement turned out to be due to a theoretical mistake. The $g - 2$ calculation is particularly tricky, not so much because it involves thousands of Feynman diagrams (this was systematized long ago) but because it requires combining effects from several different sources: QED, the weak interactions, and effects of virtual hadrons, in particular an effect called “hadronic light by light scattering.” This last effect cannot be measured, nor can it be calculated from first principles; it is calculated using phenomenological models of hadrons.

Being something of a weak link, this part of the calculation was scrutinized particularly carefully, with several groups each using their own preferred models. Thus it was a surprise when a mistake was discovered.

Clearly comparing experiment and theory in the ninth decimal place is a tricky business. And this is not the first time $g - 2$ has disagreed between theory and experiment. For several years (1996-2001) there was also a 3 sigma disagreement, inspiring many speculations about new physics.

This disagreement turned out to be due to a theoretical mistake. The $g - 2$ calculation is particularly tricky, not so much because it involves thousands of Feynman diagrams (this was systematized long ago) but because it requires combining effects from several different sources: QED, the weak interactions, and effects of virtual hadrons, in particular an effect called “hadronic light by light scattering.” This last effect cannot be measured, nor can it be calculated from first principles; it is calculated using phenomenological models of hadrons.

Being something of a weak link, this part of the calculation was scrutinized particularly carefully, with several groups each using their own preferred models. Thus it was a surprise when a mistake was discovered.

Comment on the sign of the pseudoscalar pole contribution to the muon g-2

Masashi Hayakawa, Toichiro Kinoshita

(Submitted on 6 Dec 2001 (v1), last revised 16 Dec 2001 (this version, v2))

We correct the error in the sign of the pseudoscalar pole contribution to the muon g-2, which dominates the $O(\alpha^3)$ hadronic light-by-light scattering effect. The error originates from our oversight of a feature of the algebraic manipulation program FORM which defines the epsilon-tensor in such a way that it satisfies the relation $\epsilon_{\mu_1 \mu_2 \mu_3 \mu_4} \epsilon_{\nu_1 \nu_2 \nu_3 \nu_4} \eta^{\mu_1 \nu_1} \eta^{\mu_2 \nu_2} \eta^{\mu_3 \nu_3} \eta^{\mu_4 \nu_4} = 24$, irrespective of space-time metric. To circumvent this problem, we replaced the product $\epsilon_{\mu_1 \mu_2 \mu_3 \mu_4} \epsilon_{\nu_1 \nu_2 \nu_3 \nu_4}$ by $-\eta_{\mu_1 \nu_1} \eta_{\mu_2 \nu_2} \eta_{\mu_3 \nu_3} \eta_{\mu_4 \nu_4} \text{pm} \cdot$ in the FORM-formatted program, and obtained a positive value for the pseudoscalar pole contribution, in agreement with the recent result obtained by Knecht *et al.*

Comments: 7 pages, LaTeX 2epsilon

Subjects: **High Energy Physics – Phenomenology (hep-ph)**

Report number: KEK-TH-793

Cite as: [arXiv:hep-ph/0112102](#)

(or [arXiv:hep-ph/0112102v2](#) for this version)

Submission history

From: Masashi Hayakawa [[view email](#)]

[v1] Thu, 6 Dec 2001 20:19:12 UTC (8 KB)

[v2] Sun, 16 Dec 2001 12:13:36 UTC (8 KB)

Which authors of this paper are endorsers? | [Disable MathJax](#) ([What is MathJax?](#))

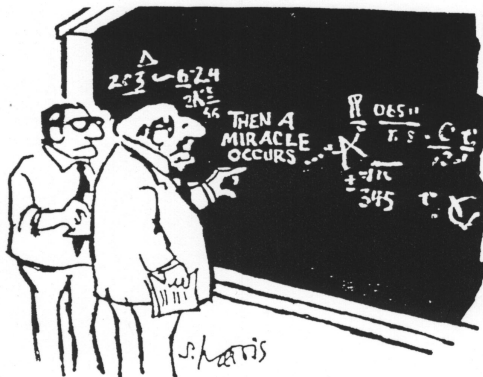
Such mistakes are of course not infrequent, even in the published literature. This one was exceptional only in the long time it took to discover it. Could this mistake, or the inevitable future mistakes, have been caught by any sort of formal verification?

It's not clear to me, but perhaps the answer is yes. But the challenge is that the mistake did not come from any single theoretical assumption or calculation, rather it came at the step of integrating various theoretical subresults derived using different physical subtheories and approximations, and not noticing a difference in conventions in one of the results. So one suspects that it would only be caught by a “large scale” formalization that covered the problem as a whole, not the individual subtheories. And while the subtheories can to some extent be formulated in a mathematically rigorous way, the overall problem cannot.

Such mistakes are of course not infrequent, even in the published literature. This one was exceptional only in the long time it took to discover it. Could this mistake, or the inevitable future mistakes, have been caught by any sort of formal verification?

It's not clear to me, but perhaps the answer is yes. But the challenge is that the mistake did not come from any single theoretical assumption or calculation, rather it came at the step of integrating various theoretical subresults derived using different physical subtheories and approximations, and not noticing a difference in conventions in one of the results. So one suspects that it would only be caught by a “large scale” formalization that covered the problem as a whole, not the individual subtheories. And while the subtheories can to some extent be formulated in a mathematically rigorous way, the overall problem cannot.

AI and theorem proving



I think you should use deep learning, here in Step 2

Many groups are applying machine learning to make ITV more automatic, in AITP projects such as TacticToe (Gauthier, Kalisczyk and Urban 2017), GamePad (Huang, Dhariwal, Song and Sutskever 2018), HOList (Bansal, Loos, Rabe, Szegedy and Wilcox 2019), CoqGym (Yang and Deng, 2019), and GPT-F (Han *et al*, 2021).

In developing a proof, many choices must be made, including

- Premise selection. Out of the many known true statements, which ones should be used to make the next deduction? There could be 100's of candidates in the current context, and if we search the entire library of proved theorems, millions of candidates.
- Tactic selection. Tactics include introduction of antecedent clauses, rewriting and simplification, and other simple logical steps. Modern ITV's typically provide 40–100 tactics.

We can make an analogy between these successive choices, and the moves in a game of solitaire.

Many groups are applying machine learning to make ITV more automatic, in AITP projects such as TacticToe (Gauthier, Kalisczyk and Urban 2017), GamePad (Huang, Dhariwal, Song and Sutskever 2018), HOList (Bansal, Loos, Rabe, Szegedy and Wilcox 2019), CoqGym (Yang and Deng, 2019), and GPT-F (Han *et al*, 2021).

In developing a proof, many choices must be made, including

- Premise selection. Out of the many known true statements, which ones should be used to make the next deduction? There could be 100's of candidates in the current context, and if we search the entire library of proved theorems, millions of candidates.
- Tactic selection. Tactics include introduction of antecedent clauses, rewriting and simplification, and other simple logical steps. Modern ITV's typically provide 40–100 tactics.

We can make an analogy between these successive choices, and the moves in a game of solitaire.

A very successful approach to AI game play is reinforcement learning (RL), as famously used by AlphaGo. Its central parts are a pair of neural networks, one to choose moves and the other to evaluate the score of game positions. The original AlphaGo used a corpus of human games for initial training, and then generated games by self-play. (Another important element is Monte Carlo tree search – it turns out for Go that playing out a game many times with random moves, gives a good estimate for its score.)

Similarly, the AITP systems use a corpus of proven theorems as training data. As the verifier works through a theorem, each step of premise and tactic selection is saved, along with a summary of the state just before the choice is made. One can then use these pairs (state, selection) to train networks to do premise and tactic selection.

The better developed systems (Coq, HOL) have large libraries with 30,000–70,000 (short) theorems. This is enough training data to achieve success rates in proving similar theorems (a held-out testing set) of around 75% (it was 50% two years ago).

A very successful approach to AI game play is reinforcement learning (RL), as famously used by AlphaGo. Its central parts are a pair of neural networks, one to choose moves and the other to evaluate the score of game positions. The original AlphaGo used a corpus of human games for initial training, and then generated games by self-play. (Another important element is Monte Carlo tree search – it turns out for Go that playing out a game many times with random moves, gives a good estimate for its score.)

Similarly, the AITP systems use a corpus of proven theorems as training data. As the verifier works through a theorem, each step of premise and tactic selection is saved, along with a summary of the state just before the choice is made. One can then use these pairs (state, selection) to train networks to do premise and tactic selection.

The better developed systems (Coq, HOL) have large libraries with 30,000–70,000 (short) theorems. This is enough training data to achieve success rates in proving similar theorems (a held-out testing set) of around 75% (it was 50% two years ago).

A very successful approach to AI game play is reinforcement learning (RL), as famously used by AlphaGo. Its central parts are a pair of neural networks, one to choose moves and the other to evaluate the score of game positions. The original AlphaGo used a corpus of human games for initial training, and then generated games by self-play. (Another important element is Monte Carlo tree search – it turns out for Go that playing out a game many times with random moves, gives a good estimate for its score.)

Similarly, the AITP systems use a corpus of proven theorems as training data. As the verifier works through a theorem, each step of premise and tactic selection is saved, along with a summary of the state just before the choice is made. One can then use these pairs (state, selection) to train networks to do premise and tactic selection.

The better developed systems (Coq, HOL) have large libraries with 30,000–70,000 (short) theorems. This is enough training data to achieve success rates in proving similar theorems (a held-out testing set) of around 75% (it was 50% two years ago).

Auto-formalization

The total corpus of mathematical texts is much larger of course – there are about 1.5 million papers just on arXiv. Building on the success of neural networks for machine translation, could we develop a system which translates “informal” mathematical text to formal mathematics?

These experiments are in very early days, see for example <https://arxiv.org/abs/1611.09703> by Kaliszyk *et al.* One problem is that there is no sizable corpus of aligned informal and formal mathematics to use as training data. So far this is dealt with by “informalizing” a formal corpus.

To my mind, a deeper problem is that mathematical texts are almost never self-contained, and a formalization cannot make much sense without the formal definitions of the concepts it refers to. In the best cases a text will only refer to standard concepts, so having a library as in Formal Abstracts would be a great help. But in many (most?) cases research papers refer to other research papers,

Auto-formalization

The total corpus of mathematical texts is much larger of course – there are about 1.5 million papers just on arXiv. Building on the success of neural networks for machine translation, could we develop a system which translates “informal” mathematical text to formal mathematics?

These experiments are in very early days, see for example <https://arxiv.org/abs/1611.09703> by Kaliszyk *et al.* One problem is that there is no sizable corpus of aligned informal and formal mathematics to use as training data. So far this is dealt with by “informalizing” a formal corpus.

To my mind, a deeper problem is that mathematical texts are almost never self-contained, and a formalization cannot make much sense without the formal definitions of the concepts it refers to. In the best cases a text will only refer to standard concepts, so having a library as in Formal Abstracts would be a great help. But in many (most?) cases research papers refer to other research papers,

Auto-formalization

The total corpus of mathematical texts is much larger of course – there are about 1.5 million papers just on arXiv. Building on the success of neural networks for machine translation, could we develop a system which translates “informal” mathematical text to formal mathematics?

These experiments are in very early days, see for example <https://arxiv.org/abs/1611.09703> by Kaliszyk *et al.* One problem is that there is no sizable corpus of aligned informal and formal mathematics to use as training data. So far this is dealt with by “informalizing” a formal corpus.

To my mind, a deeper problem is that mathematical texts are almost never self-contained, and a formalization cannot make much sense without the formal definitions of the concepts it refers to. In the best cases a text will only refer to standard concepts, so having a library as in Formal Abstracts would be a great help. But in many (most?) cases research papers refer to other research papers.

Reinforcement learning

The big success of AlphaGo came from its ability to generate its own training data by self-play. Amazingly, it turned out that human games were not needed – AlphaGo Zero achieved superhuman skill without them. Could we make a theorem prover do self-play?

At first, one might say that to “win a game” is equivalent to proving a theorem. However this is simplistic as every step of a deduction proves a new logical statement. Somehow the results have to be scored according to how “significant,” “interesting” or “useful” the statement is, or how close the new statement is to a significant result.

Only rewarding the theorems considered interesting or significant by humans may not be giving the computer enough feedback. So, it may be necessary to give the computer its own ability to judge what is interesting. From an ML point of view this is just another scoring function which could be learned.

Reinforcement learning

The big success of AlphaGo came from its ability to generate its own training data by self-play. Amazingly, it turned out that human games were not needed – AlphaGo Zero achieved superhuman skill without them. Could we make a theorem prover do self-play?

At first, one might say that to “win a game” is equivalent to proving a theorem. However this is simplistic as every step of a deduction proves a new logical statement. Somehow the results have to be scored according to how “significant,” “interesting” or “useful” the statement is, or how close the new statement is to a significant result.

Only rewarding the theorems considered interesting or significant by humans may not be giving the computer enough feedback. So, it may be necessary to give the computer its own ability to judge what is interesting. From an ML point of view this is just another scoring function which could be learned.

Reinforcement learning

The big success of AlphaGo came from its ability to generate its own training data by self-play. Amazingly, it turned out that human games were not needed – AlphaGo Zero achieved superhuman skill without them. Could we make a theorem prover do self-play?

At first, one might say that to “win a game” is equivalent to proving a theorem. However this is simplistic as every step of a deduction proves a new logical statement. Somehow the results have to be scored according to how “significant,” “interesting” or “useful” the statement is, or how close the new statement is to a significant result.

Only rewarding the theorems considered interesting or significant by humans may not be giving the computer enough feedback. So, it may be necessary to give the computer its own ability to judge what is interesting. From an ML point of view this is just another scoring function which could be learned.

What is “interesting” mathematics? In Lenat’s AM system (1976) this was defined by hand-coded heuristics.

Perhaps given a large semi-formal corpus, this could be inferred by similarity of a statement to the statements in the corpus. After all, people try to make interesting statements and avoid uninteresting ones.

Rather than make an *a priori* definition of interesting, one can say that an interesting concept is one which aids reasoning. To the extent that the system can judge the complexity of its proofs, then a new statement which makes many proofs simpler is *ipso facto* interesting.

One could consider efficacy at more general tasks. Perhaps textbook problems would be a good source. As another example, given a mathematical definitions such as “finite group,” can the system take a pair of randomly chosen examples and efficiently prove that they are isomorphic or not isomorphic. As an even harder test, can the system enumerate groups with up to k elements?

What is “interesting” mathematics? In Lenat’s AM system (1976) this was defined by hand-coded heuristics.

Perhaps given a large semi-formal corpus, this could be inferred by similarity of a statement to the statements in the corpus. After all, people try to make interesting statements and avoid uninteresting ones.

Rather than make an *a priori* definition of interesting, one can say that an interesting concept is one which aids reasoning. To the extent that the system can judge the complexity of its proofs, then a new statement which makes many proofs simpler is *ipso facto* interesting.

One could consider efficacy at more general tasks. Perhaps textbook problems would be a good source. As another example, given a mathematical definitions such as “finite group,” can the system take a pair of randomly chosen examples and efficiently prove that they are isomorphic or not isomorphic. As an even harder test, can the system enumerate groups with up to k elements?

What is “interesting” mathematics? In Lenat’s AM system (1976) this was defined by hand-coded heuristics.

Perhaps given a large semi-formal corpus, this could be inferred by similarity of a statement to the statements in the corpus. After all, people try to make interesting statements and avoid uninteresting ones.

Rather than make an *a priori* definition of interesting, one can say that an interesting concept is one which aids reasoning. To the extent that the system can judge the complexity of its proofs, then a new statement which makes many proofs simpler is *ipso facto* interesting.

One could consider efficacy at more general tasks. Perhaps textbook problems would be a good source. As another example, given a mathematical definitions such as “finite group,” can the system take a pair of randomly chosen examples and efficiently prove that they are isomorphic or not isomorphic. As an even harder test, can the system enumerate groups with up to k elements?

Summary and conclusions

Some major lines of progress in computers and mathematics:

- General advances in hardware and software
- More and larger knowledge repositories, with better metadata
- Neural networks and deep learning
- Interactive theorem verification, theorem libraries and formal abstracts
- New styles of collaboration along the lines of the Stack project, Polymath, ...

Speculations and opportunities

In the end I don't think there are very clear predictions to make for 2030, but let me go out on a limb and make some anyways.

- There will be textbooks which integrate computation – not just numerics but many of the new technologies we discuss – in groundbreaking ways, making today's textbooks look dated.
- Significant new discoveries and proofs will continue to be made using computational experiment and numerical methods, but of the general character we have seen: solving intricate problems by combinatorial search, producing large datasets leading to statistical conjectures, *etc.*
- ITVs will be much easier to use, thanks to AI automation. An introductory course in theorem proving will be a common advanced undergraduate offering. They will have a status much like computer algebra systems now – a convenient tool that some people rely on and many people use (say) once a month.

Speculations and opportunities

In the end I don't think there are very clear predictions to make for 2030, but let me go out on a limb and make some anyways.

- There will be textbooks which integrate computation – not just numerics but many of the new technologies we discuss – in groundbreaking ways, making today's textbooks look dated.
- Significant new discoveries and proofs will continue to be made using computational experiment and numerical methods, but of the general character we have seen: solving intricate problems by combinatorial search, producing large datasets leading to statistical conjectures, *etc.*
- ITVs will be much easier to use, thanks to AI automation. An introductory course in theorem proving will be a common advanced undergraduate offering. They will have a status much like computer algebra systems now – a convenient tool that some people rely on and many people use (say) once a month.

Speculations and opportunities

In the end I don't think there are very clear predictions to make for 2030, but let me go out on a limb and make some anyways.

- There will be textbooks which integrate computation – not just numerics but many of the new technologies we discuss – in groundbreaking ways, making today's textbooks look dated.
- Significant new discoveries and proofs will continue to be made using computational experiment and numerical methods, but of the general character we have seen: solving intricate problems by combinatorial search, producing large datasets leading to statistical conjectures, *etc.*
- ITVs will be much easier to use, thanks to AI automation. An introductory course in theorem proving will be a common advanced undergraduate offering. They will have a status much like computer algebra systems now – a convenient tool that some people rely on and many people use (say) once a month.

Speculations and opportunities

In the end I don't think there are very clear predictions to make for 2030, but let me go out on a limb and make some anyways.

- There will be textbooks which integrate computation – not just numerics but many of the new technologies we discuss – in groundbreaking ways, making today's textbooks look dated.
- Significant new discoveries and proofs will continue to be made using computational experiment and numerical methods, but of the general character we have seen: solving intricate problems by combinatorial search, producing large datasets leading to statistical conjectures, *etc.*
- ITVs will be much easier to use, thanks to AI automation. An introductory course in theorem proving will be a common advanced undergraduate offering. They will have a status much like computer algebra systems now – a convenient tool that some people rely on and many people use (say) once a month.

- A large database of formal mathematics will exist, built by some combination of human and automated work. Perhaps the overall organization will be created or tuned by humans, while the bulk of the formalization will be automatic.
- Semantic mathematical search will be a standard part of our literature searches. At least one nontrivial connection between different mathematical fields will be discovered this way.
- At present it is hard to reuse code from math/physics projects – this problem will be largely solved.

But in 2030 will computers have invented or proven any major result by themselves? Math AGI, either based on the human mathematical literature, or which trains itself *ab initio*, is an old dream.

- A large database of formal mathematics will exist, built by some combination of human and automated work. Perhaps the overall organization will be created or tuned by humans, while the bulk of the formalization will be automatic.
- Semantic mathematical search will be a standard part of our literature searches. At least one nontrivial connection between different mathematical fields will be discovered this way.
- At present it is hard to reuse code from math/physics projects – this problem will be largely solved.

But in 2030 will computers have invented or proven any major result by themselves? Math AGI, either based on the human mathematical literature, or which trains itself *ab initio*, is an old dream.

- A large database of formal mathematics will exist, built by some combination of human and automated work. Perhaps the overall organization will be created or tuned by humans, while the bulk of the formalization will be automatic.
- Semantic mathematical search will be a standard part of our literature searches. At least one nontrivial connection between different mathematical fields will be discovered this way.
- At present it is hard to reuse code from math/physics projects – this problem will be largely solved.

But in 2030 will computers have invented or proven any major result by themselves? Math AGI, either based on the human mathematical literature, or which trains itself *ab initio*, is an old dream.

- A large database of formal mathematics will exist, built by some combination of human and automated work. Perhaps the overall organization will be created or tuned by humans, while the bulk of the formalization will be automatic.
- Semantic mathematical search will be a standard part of our literature searches. At least one nontrivial connection between different mathematical fields will be discovered this way.
- At present it is hard to reuse code from math/physics projects – this problem will be largely solved.

But in 2030 will computers have invented or proven any major result by themselves? Math AGI, either based on the human mathematical literature, or which trains itself *ab initio*, is an old dream.

While progress in AI is rapid, we don't have a good way to guess how far away this dream might be. But here are some thoughts:

- Math reasoning – at the level of individual steps – is not likely to be easier than general reasoning in other large domains.
- Other domains may have advantages, such as more training data.
- The advantages of math as a domain are that one can use arbitrarily long chains of reasoning, and that success depends much less on abilities other than logical reasoning. So, even if general reasoning capability is developed in many domains at once, it will have major consequences in math before most fields.
- Significant advances are being made in general reasoning, and breakthroughs may happen soon.
- The full consequences of a breakthrough will take around ten years to realize. For example, MCTS was introduced in computer Go around 2006, and produced significant improvements by 2009, leading to AlphaGo in 2016.