Automated Reasoning and Symbolic AI

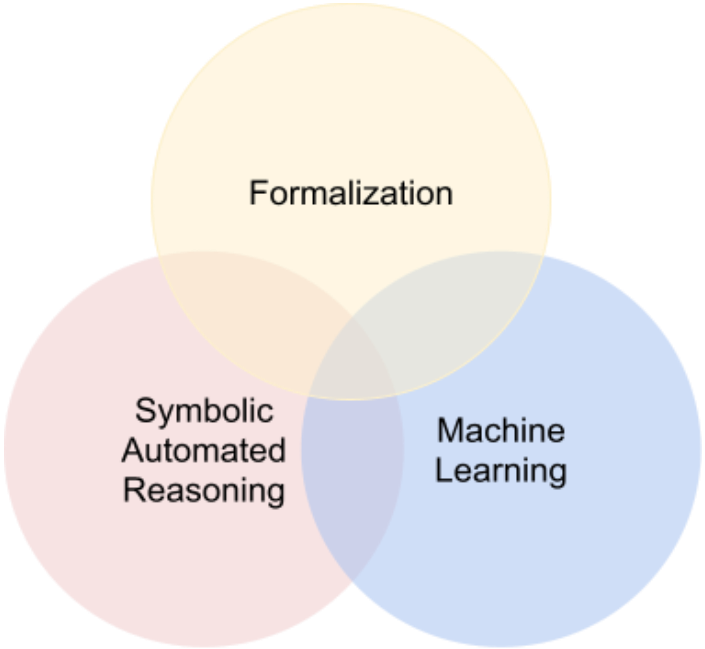
Jeremy Avigad
Department of Philosophy
Department of Mathematical Sciences
Carnegie Mellon University

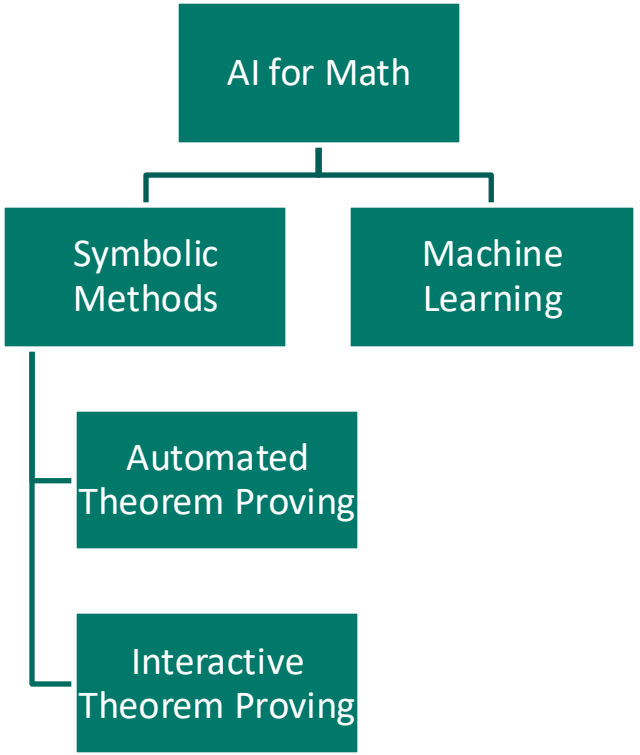
Institute for Computer-Aided Reasoning in Mathematics

October 15, 2025

Overview

AI for mathematics





Automated reasoning

Two types of logic-based reasoning algorithms:

- **Decision procedures:**
  - Q: Is X true? Is X provable?
  - return yes or no
- **Search procedures:**
  - Q: Is X true? Is X provable?
  - Search for justification for a "yes" answer

Decision procedures can be very inefficient. Undecidability sets in quickly.

In practice, systems can say “yes,” “no,” “I don’t know,” or run forever.

Propositional logic

A propositional formula is

- **Valid** (a **tautology**) if it is true under every truth assignment
- **Satisfiable** if it is true under some truth assignment
- **Unsatisfiable** if is it false under every truth assignment

P	Q	$P \rightarrow Q$	$(P \rightarrow Q) \wedge Q$	$(P \rightarrow Q) \wedge Q \rightarrow P$
T	T	T	T	T
T	F	F	F	T
F	T	T	T	F
F	F	T	F	T

Propositional logic

Propositional logic is decidable (in fact, NP-complete).

Modern SAT solvers can decide industrial formulas with tens of millions of variables and hundreds of millions of clauses, often in minutes.

If the formula is satisfiable, you get a satisfying assignment. If the formula is unsatisfiable, you get an independently checkable proof.

Recipe for mathematics:

- Encode / reduce a problem to a SAT problem.
- Use a SAT solver.

First-order logic

Adds quantifiers:

$$Even(x) \equiv \exists y(x = y + y)$$

$$Prime(x) \equiv x \geq 2 \wedge \forall y > 0(y|x \rightarrow y = 1 \vee y = x)$$

$$Goldbach \equiv \forall x\ (Even(x) \wedge x > 2\ \rightarrow \exists y, z\ (Prime(y)\ \wedge Prime(z)\ \wedge x = y + z)$$

The family of questions "Is X provable from axioms A?" is generally equivalent to the halting problem.

The family of questions "Is X true?" is generally even more undecidable.

Decidability and undecidability

Other decidable problems:

- real linear arithmetic (with equations and inequalities)
- integer linear arithmetic
- real closed fields
- algebraically closed fields

Equivalent to the halting problem:

- provability in first-order logic (without axioms)
- provability from axioms that include some basic arithmetic
- provability from axioms that are consistent with an interpretation of arithmetic

Automation

Three important families:

- **SAT solvers:** decision procedures for propositional logic
- **First-order provers:** search procedures for proofs in first-order logic
- **SMT solvers:** combine decision procedures
  - propositional logic
  - equational logic
  - linear integer arithmetic
  - linear real arithmetic

  with some search as well

SMT solvers play an important role in software verification.

History

Origins of automated reasoning

Early contributions:

- Martin Davis implemented Presburger's decision procedure at the IAS in 1954.
- Allen Newell, Herbert Simon, and Cliff Shaw introduced the *Logic Theorist* in 1956.
- Hao Wang implemented good provers for propositional and predicate logic in 1958.
- Henry Gelernter, J. R. Hansen, and Donald Loveland published an article on the *Geometry Machine* in 1960.
- Davis and Hilary Putnam introduced the propositional resolution rule in 1960.
- John Alan Robinson introduced a unification algorithm in 1965.

Origins of automated reasoning

The first incompleteness theorem applies to any consistent, computably axiomatized theory containing basic arithmetic.

Gödel, 1931: The theorem “is not in any way due to the special nature of the systems that have been set up, but holds for a wide class of formal systems; among these, in particular, are all systems that result from the two just mentioned through the addition of a finite number of axioms…”

He gave a tentative definition of computability at the IAS in 1932.

Origins of automated reasoning

After Turing's 1936 paper, he added this footnote:

“In consequence of later advances, in particular of the fact that, due to A. M. Turing's work, a precise and unquestionably adequate definition of the general concept of a formal system can now be given, the existence of undecidable arithmetical propositions and the non-demonstrability of the consistency of a system in the same system can now be proved rigorously for *every* consistent formal system containing a certain amount of finitary number theory.”

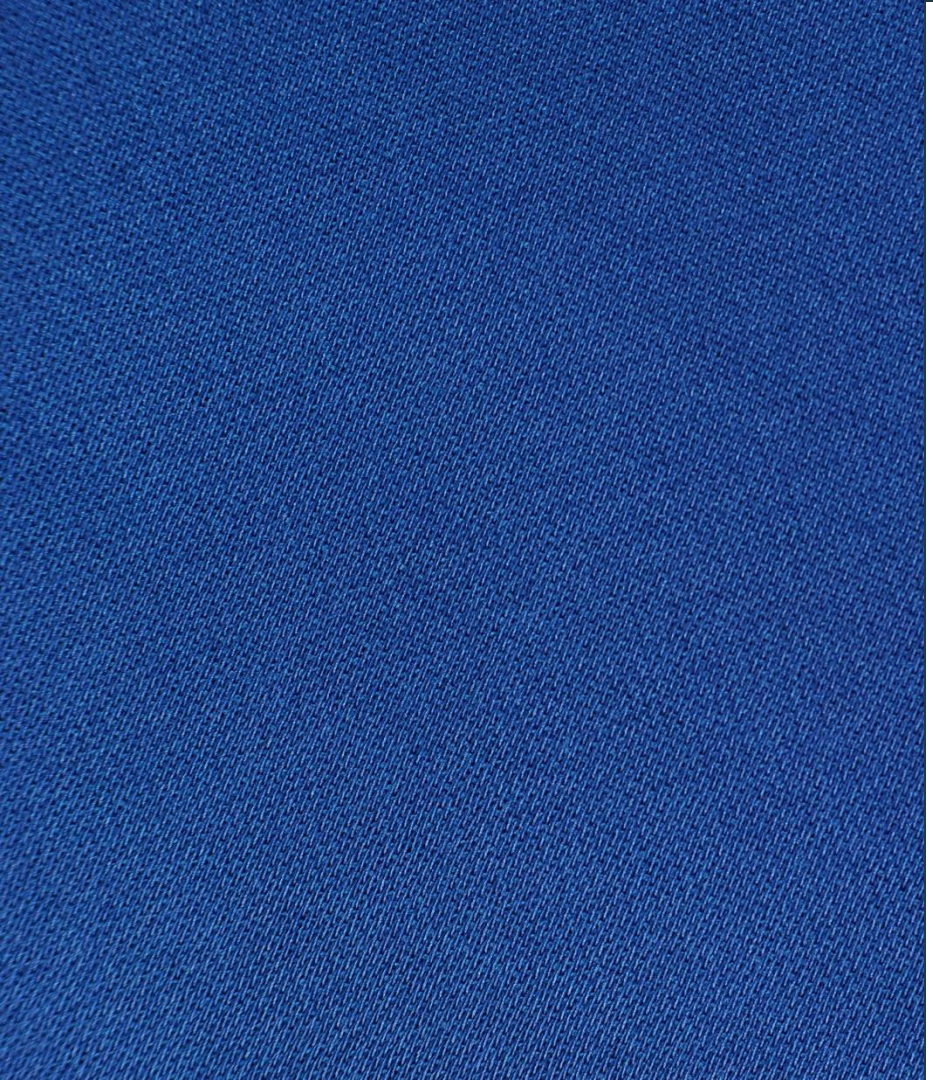
Origins of automated reasoning

More connections to automated reasoning:

- Turing presented his definition of computability in 1936 with a negative solution to the *Entscheidungsproblem*.
- He also noted that incompleteness follows from undecidability, because one can computably search for proofs.
- Church gave another proof of the undecidability of arithmetic in 1936.
- Kleene was also keenly interested in logic and foundations.

The origins of decision procedures are even earlier:

- In 1915, Löwenheim proved the decidability of monadic first-order logic.
- Presburger presented his decision procedure for arithmetic in 1929.
- Tarski had a decision procedure for real closed fields in 1930.



SAT Solvers

SAT solvers

SAT solvers accept formulas in *Conjunctive Normal Form* (CNF), such as:

$$
\begin{gathered}
P \ \wedge \\
(\neg P \vee Q) \ \wedge \\
(\neg Q \vee R) \wedge \\
\neg R
\end{gathered}
$$

Any propositional is *equivalent* to one in CNF, and *equisatisfiable with* one in CNF that is not much longer.

This particular formula is UNSAT.

Ramsey's theorem

Theorem (Ramsey). For every c and k, there is an n large enough, such that for every c-coloring of the edges of the complete graph on n vertices, there is a monochromatic clique of size k.

Fix c and n. To describe a c-coloring of the edges, use variables $P_{i,j,u}$, where $i < j < n$, which says that the edge from i to j gets color u.

- Every edge gets a color: $\bigvee_u P_{i,j,u}$ for each $i < j$.
- No edge gets more than one color: $\neg P_{i,j,u} \vee \neg P_{i,j,v}$ for $u \neq v$.
- No monochromatic clique: $\bigvee_{i,j \in C} \neg P_{i,j,u}$ for each clique of size k.

The happy ending problem

Theorem (Erdős and Szekeres, 1935). For any k there is an n large enough such that any n points in general position (no three colinear) contain a convex k-gon.

Let $f(k)$ be the least such n.

Then, for example, $f(4) = 5$.

We know $f(5) = 9$. and $f(6) = 17$. The value of $f(7)$ is not known.

Empty convex polygons

There are infinite sets of points in general position with no *empty* 7-gon.

But Heule and Scheucher recently showed that every set of 30 points contains an empty hexagon, and this is sharp.

Computer Science > Computational Geometry

[Submitted on 1 Mar 2024]

Happy Ending: An Empty Hexagon in Every Set of 30 Points

Marijn J.H. Heule, Manfred Scheucher

Satisfiability solving has been used to tackle a range of long-standing open math problems in recent years. We add another success by solving a geometry problem that originated a century ago. In the 1930s, Esther Klein's exploration of unavoidable shapes in planar point sets in general position showed that every set of five points includes four points in convex position. For a long time, it was open if an empty hexagon, i.e., six points in convex position without a point inside, can be avoided. In 2006, Gerken and Nicolás independently proved that the answer is no. We establish the exact bound: Every 30-point set in the plane in general position contains an empty hexagon. Our key contributions include an effective, compact encoding and a search-space partitioning strategy enabling linear-time speedups even when using thousands of cores.

Units in group rings

If $G$is a group and R is a ring, the *group ring* $R[G]$ is the ring of expressions

$$r_1 g_1 + \cdots + r_n g_n$$

where each $r_i \in R, g_i \in G$, with the natural addition and multiplication.

In his 1940 thesis, Graham Higman conjectured what has become to be known as the Kaplansky Unit Conjecture: if G is torsion-free and K is a field, then $K[G]$ has no nontrivial units.

In an *Annals* paper in 2021, Giles Gardam provided a counterexample. He used a SAT solver to find it.

A counterexample to the unit conjecture for group rings

By Giles Gardam

To the memory of Willem Henskens

Abstract

The unit conjecture, commonly attributed to Kaplansky, predicts that if K is a field and G is a torsion-free group, then the only units of the group ring $K[G]$ are the trivial units, that is, the non-zero scalar multiples of group elements. We give a concrete counterexample to this conjecture; the group is virtually abelian and the field is order two.

THEOREM A. *Let P be the torsion-free group defined by the presentation $\langle a, b \,|\, b^{-1}a^2b = a^{-2},\ a^{-1}b^2a = b^{-2} \rangle$, and set $x = a^2, y = b^2, z = (ab)^2$. Set*

$$p = (1+x)(1+y)(1+z^{-1}), \qquad q = x^{-1}y^{-1} + x + y^{-1}z + z,$$
$$r = 1 + x + y^{-1}z + xyz, \qquad s = 1 + (x + x^{-1} + y + y^{-1})z^{-1}.$$

Then $p + qa + rb + sab$ is a non-trivial unit in the group ring $\mathbb{F}_2[P]$.

Objection

SAT solvers are good at

- ruling out finite configurations
- finding finite objects (often counterexamples)

Doesn’t this limit their utility for doing *real* mathematics?

Response: they provide very general means of efficient combinatorial search.

We need to learn how to use them for infinitary problems and constructions.

SAT and ITP

Verifying SAT results

Steps:

- Reduce a mathematical problem to a combinatorial problem
- Encode the combinatorial problems as a SAT problem
- Use clever ideas to break symmetries and reduce the search space.
- Run a SAT solver
  - SAT result: translate back to mathematical object
  - UNSAT: declare "the theorem is true"

Should mathematicians trust this?

It helps that SAT solvers can emit independently checkable proofs, and there are verified proof checkers for several formats.

Keller's conjecture

Recall Keller's conjecture from the first lecture.

Joshua Clune verified the pen-and-paper reduction of the geometric problem to a graph-theoretic problem, in Lean 3.

James Gallicchio:

- Reproved the reduction in Lean 4.
- Verified the translation of the graph-theoretic statement to a SAT formula in Lean 4, including symmetry-breaking reductions.
- Encoded additional symmetry-breaking rules in a proof format known as SR.
- Used an SR checker, verified in Lean 4 by Cayden Codel, to verify proofs generated by a SAT solver.

The empty hexagon problem

The verification of the claim that every set of 30 points in general position required clever encodings of the geometric data and symmetry breaking.

Every triple of points gives rise to an orientation (turn left / turn right) and the solution involves ruling out configurations based on properties of orientations.

The symmetry-breaking is equally complex.

The reduction of the problem to a SAT formula was verified in Lean by Bernardo Subercaseaux, Wojciech Nawrocki, James Gallicchio, Cayden Codel, Mario Carneiro, and Marijn Heule.

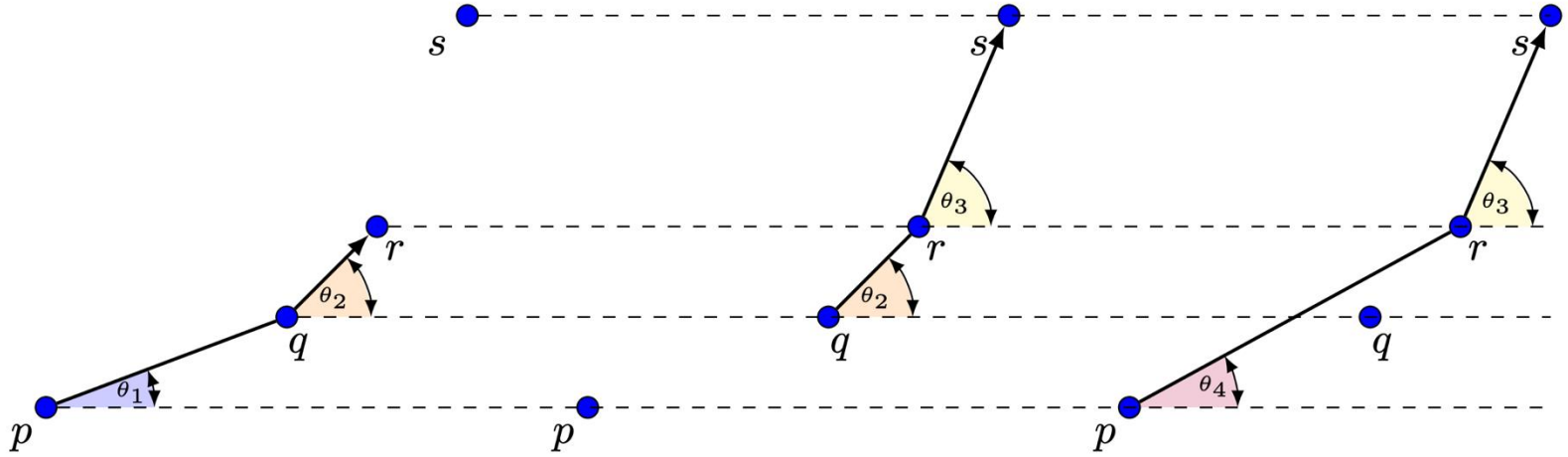
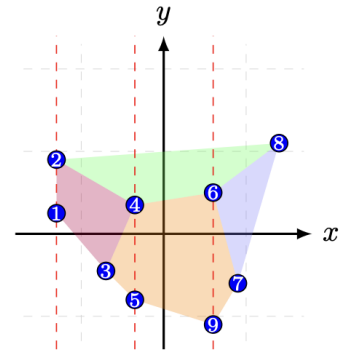
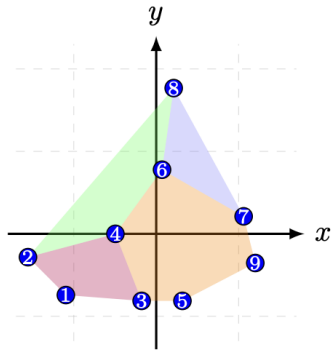


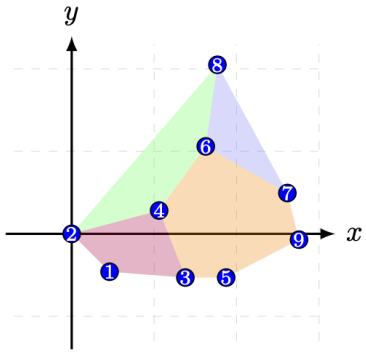
Figure 3 Illustration for $\sigma(p, q, r) = 1 \wedge \sigma(q, r, s) = 1 \implies \sigma(p, r, s) = 1$. As we have assumptions $\theta_3 > \theta_2 > \theta_4$ by the forward direction of the *slope-orientation equivalence*, we deduce $\theta_3 > \theta_4$, and then conclude $\sigma(p, r, s) = 1$ by the backward direction of the *slope-orientation equivalence*.



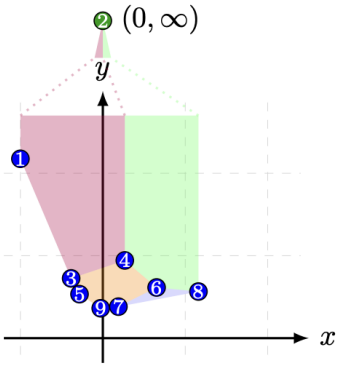
(a) The original list of points.



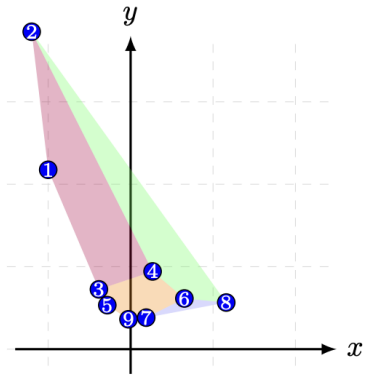
(b) There always exists a rotation (in this case by $45°$) that makes all the x-coordinates different.



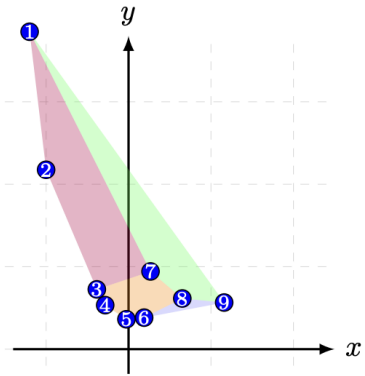
(c) After translating, the leftmost point is at $(0, 0)$.



(d) Result after applying the map $(x, y) \mapsto (y/x, 1/x)$.



(e) Point 2 is brought back into the real plane.



(f) Points are relabeled from left to right.

Figure 5 Illustration of the proof of the symmetry breaking theorem. Note that the highlighted holes are preserved as σ-equivalence is preserved. For simplicity we have omitted the illustration of the Lex order property.

First-order provers

Early applications

In 1996, William McCune used an equational theorem prover to prove the Robbins conjecture, which states that a certain system of equations axiomatizes Boolean algebras.

McCune showed that $\left(w\left((x^{-1}w)^{-1}z\right)\right)\left((yz)^{-1}y\right) = x$ axiomatizes groups.

Kenneth Kunen showed that this is the shortest such axiom.

In the 90s and 00s, people used first-order provers to study exotic structures like loops and quasigroups.

Science Times

The New York Times

TUESDAY, DECEMBER 10, 1996

ors Decide on Is Key uts' Health

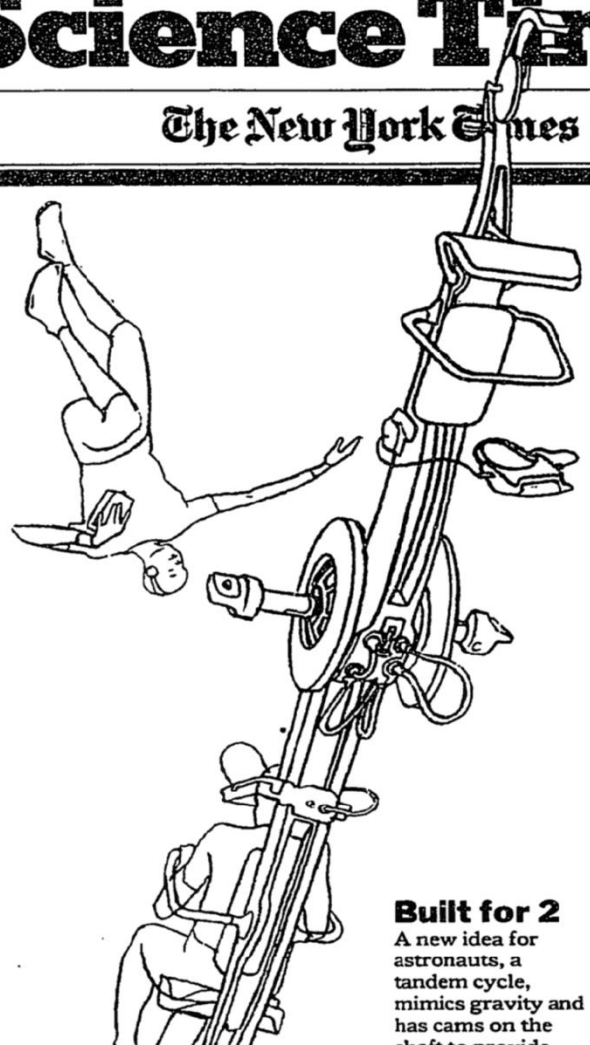
from the hip and lower spine, a trend that if uncorrected over time could prevent long space voyages.

Experts say a trip to Mars, a year or two each way, carries the risk of leaving an astronaut crippled upon return.

"We've learned that bone loss from selected sites on the skeleton is a problem that we still don't have a solution to," Dr. Frank M. Sulzman, director of life science research at the National Aeronautics and Space Administration, said in an interview.

But NASA and its advisers say they are on the verge of finding what may be a simple way to prevent a wide range of space illnesses: nothing fancy or high-tech, it boils down to hard exercise, the orbital equivalent of pumping iron.

Astronauts now tend to do endurance types of exercise, including cycling, rowing and walking on a treadmill, that stress aerobics and stamina. But a wide consensus is developing among space physiologists and NASA officials that this approach is wrong and needs to be supplemented by strenuous workouts that increase



Built for 2
A new idea for astronauts, a tandem cycle, mimics gravity and has cams on the

With Major Math Proof, Brute Computers Show Flash of Reasoning Power

The achievement would have been called creative if a human had done it.

By GINA KOLATA

COMPUTERS are whizzes when it comes to the grunt work of mathematics. But for creative and elegant solutions to hard mathematical problems, nothing has been able to beat the human mind. That is, perhaps, until now.

A computer program written by researchers at Argonne National Laboratory in Illinois has come up with a major mathematical proof that would have been called creative if a human had thought of it. In doing so, the computer has, for the first time, got a toehold into pure mathematics, a field described by its practitioners as more of an art form than a science. And the implications, some say, are profound, showing just how powerful computers can be at reasoning itself, at mimicking the flashes of logical insight or even

those conjectures were easy to prove. The difference this time is that the computer has solved a conjecture that stumped some of the best mathematicians for 60 years. And it did so with a program that was designed to reason, not to solve a specific problem. In that sense, the program is very different from chess-playing computer programs, for example, which are intended to solve just one problem: the moves of a chess game.

"It's a sign of power, of reasoning power," said Dr. Larry Wos, the supervisor of the computer reasoning project at Argonne. And with this result, obtained by a colleague, Dr. William McCune, he said, "We've taken a quantum leap forward."

Dr. Wos predicts that the result may mark the beginning of the end for mathematics research as it is now practiced, eventually freeing mathematicians to focus on discovering new conjectures, and leaving the proof to computers.

But the result also may challenge the very notion of creative thinking, raising the possibility that computers could take a parallel path to reach the same conclusions as great human thinkers. Or it may be that since no one has any idea how humans think, the magnificent bursts of

The Equational Theories Project

There are 4694 equations between terms involving at most four instances of a binary operator.

Associativity is one example: $(x \diamond y) \diamond z = x \diamond (y \diamond z)$.

On September 24, 2024, Terence Tao launched the equational theories project to determine all entailments between them. (Later also: in finite structures.)

- Each entailment required proof.
- Non-entailment required countermodels.
- Everything had to be verified in Lean.

THE EQUATIONAL THEORIES PROJECT: ADVANCING COLLABORATIVE MATHEMATICAL RESEARCH AT SCALE

MATTHEW BOLAN, JOACHIM BREITNER, JOSE BROX, MARIO CARNEIRO, MARTIN DVORAK, ANDRÉS GOENS, AARON HILL, HARALD HUSUM, ZOLTAN KOCSIS, BRUNO LE FLOCH, LORENZO LUCCIOLI, DOUGLAS MCNEIL, ALEX MEIBURG, PIETRO MONTICONE, PACE NIELSEN, GIOVANNI PAOLINI, MARCO PETRACCI, BERNHARD REINKE, DAVID RENSHAW, MARCUS ROSSEL, CODY ROUX, JÉRÉMY SCANVIC, SHREYAS SRINIVAS, ANAND RAO TADIPATRI, TERENCE TAO, VLAD TSYRKLEVICH, DANIEL WEBER, FAN ZHENG

ABSTRACT. We report on the *Equational Theories Project* (ETP), an online collaborative pilot project to explore new ways to collaborate in mathematics with machine assistance. The project successfully determined all 22 028 942 edges of the implication graph between the 4694 simplest equational laws on magmas, by a combination of human-generated and automated proofs, all validated by the formal proof assistant language *Lean*. As a result of this project, several new constructions of magmas obeying specific laws were discovered, and several auxiliary questions were also addressed, such as the effect of restricting attention to finite magmas.

The Equational Theories Project

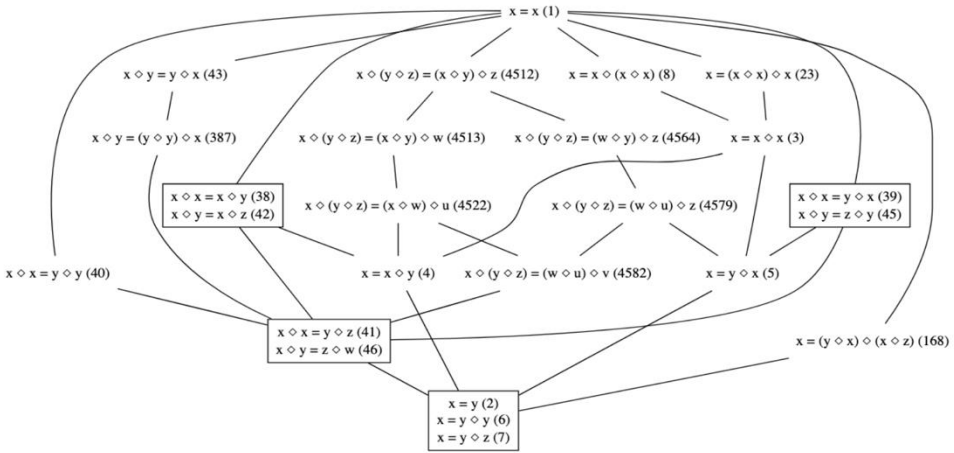
Notable features of the project:

- It was a large-scale collaboration.
- It included amateur mathematicians, computer scientists.
- It used a lot of automation, including *Vampire, Duper, Aesop, Prover9, Mace4, Z3,* and *Egg.*
- It yielded interesting new ideas, methods, insights, and results.

Equational Theories Project

Mapping out the relations between different equational theories of Magmas

Blueprint (web) | Blueprint (pdf) | Paper (pdf) | Documentation | Dashboard | Equation Explorer | Finite Magma Explorer | Graphiti | GitHub



The purpose of this project, launched on Sep 25, 2024, is to explore the space of equational theories of magmas, ordered by implication. To begin with we shall focus only on theories of a single equation, and specifically on the 4694 equational laws involving at most four magma operations, up to symmetry and relabeling (here is the list in Lean and in plain text). This creates 4694*(4694-1) = 22,028,942 implications that need to be proven or disproven, creating both “implications” and “anti-implications”.

Logic puzzles

First-order provers are good at logic puzzles.

You can find many of them in the TPTP (Thousands of Problems for Theorem Provers) library.

In 2022, Marijn Heule, Wojciech Nawrocki, and I asked students in our course, *Logic and Mechanized Reasoning,* to code up some of Smullyan's logic puzzles in Lean and send them to Vampire.

13 • The Asylum of Doctor Tarr and Professor Fether

The last asylum Craig visited he found to be the most bizarre of all. This asylum was run by two doctors named Doctor Tarr and Professor Fether. There were other doctors on the staff as well. Now, an inhabitant was called *peculiar* if he believed that he was a patient. An inhabitant was called *special* if all patients believed he was peculiar and no doctor believed he was peculiar. Inspector Craig found out that at least one inhabitant was sane and that the following condition held:

Condition C: Each inhabitant had a best friend in the asylum. Moreover, given any two inhabitants, A and B, if A believed that B was special, then A's best friend believed that B was a patient.

Shortly after this discovery, Inspector Craig had private interviews with Doctor Tarr and Professor Fether. Here is the interview with Doctor Tarr:

Craig: Tell me, Doctor Tarr, are all the doctors in this asylum sane?

Tarr: Of course they are!

Craig: What about the patients? Are they all insane?

Tarr: At least one of them is.

The second answer struck Craig as a surprisingly modest claim! Of course, if all the patients are insane, then it certainly is true that at least one is. But why was Doctor Tarr being so cautious? Craig then had his interview with Professor Fether, which went as follows:

Craig: Doctor Tarr said that at least one patient here is insane. Surely that is true, isn't it?

Professor Fether: Of course it is true! All the patients in this asylum are insane! What kind of asylum do you think we are running?

Craig: What about the doctors? Are they all sane?

Professor Fether: At least one of them is.

Craig: What about Doctor Tarr? Is he sane?

Professor Fether: Of course he is! How dare you ask me such a question?

At this point, Craig realized the full horror of the situation! What was it?

(Those who have read "The System of Doctor Tarr and Professor Fether," by Edgar Allan Poe, will probably guess the solution before they prove it is correct. See remarks following the solution.)

Logic puzzles

Smullyan proved that in the last asylum, all the doctors are insane and all the patients are insane.

Using Vampire, we showed that the assumptions are inconsistent: there is no such asylum.

We published an article, “An Impossible Asylum,” in the *American Mathematical Monthly.*

In a follow-up article, “A Possible Asylum,” Bogaerts showed that if “any two” means “any distinct two,” there are such asylums with exactly one patient.

Automated Reasoning and Interactive Theorem Proving

ATP and ITP

Disappointingly, outside of SAT solvers, the automated reasoning tools I have described have had almost no impact on mathematics.

They have had an impact on the *formalization* of mathematics:

- Decision procedures like linear arithmetic and linear integer arithmetic are commonly implemented in proof assistants.
- First-order theorem provers and SMT solvers are a key component of *sledgehammer* technology.

Sledgehammers

The task: given some hypotheses, and a conclusion, and a library with tens of thousands of theorems, construct a formal proof.

The approach:

- Use heuristics to extract a small set of promising premises from the library.
- Translate the problem to the language of one or more ATPs.
- Call external provers to prove the goal.
- If any succeeds, harvest information about the proof (possibly only the premises used).
- Use the information to reconstruct a proof internally.

The most successful one to date is Isabelle's Sledgehammer, originally developed by Larry Paulson and Jia Meng (c. 2006), and later by Jasmin Blanchette and many others.

Sledgehammers

There are variations at every step:

- One can use symbolic heuristics, lightweight ML, or neural methods for premise selection.
- One can use various methods of translation.
- One can use first-order provers or SMT solvers as external provers.
- One can harvest various types of information.
- One can use different methods of proof reconstruction.

The next slide shows the hammer web page for Isabelle 2009.

Sledgehammer

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Home
Overview
Installation
Download
Documentation
Community

Site Mirrors:
Cambridge (.uk)
Munich (.de)
Sydney (.au)

The Sledgehammer: Let Automatic Theorem Provers write your Isabelle scripts!

The sledgehammer can be used, at any point in a backward proof, with one mouse click. Your first subgoal will be converted into clause form and given to automatic theorem provers (ATPs), together with perhaps hundreds of other clauses representing currently known facts. Because jobs are run in the background, you can continue to work on your proof by other means. Provers can be run in parallel, the first successful result is displayed, and the other provers are terminated. Any reply (which may arrive minutes later) will appear in the Proof General response buffer. If a subgoal is proved, the response consists of a list of Isabelle commands to insert into the proof script. These will invoke the *Metis* prover.

Installation

Supported provers include E, SPASS, Vampire. Additionally, provers from System on TPTP can be queried over the internet. The standard Isabelle installation already includes bundled versions of E and SPASS. Remote Vampire is also preconfigured. Note that remote provers require Perl with the World Wide Web Library *libwww-perl* installed.

How to Invoke It

The sledgehammer is part of Isabelle/HOL. To call it, merely invoke the menu item *Isabelle* > *Commands* > *sledgehammer* (see screenshot). Alternatively, issue the `sledgehammer` Isar command.

For best results, first simplify your problem by calling *auto* or at least *safe* followed by *simp_all*. None of the ATPs contain arithmetic decision procedures. They are not especially good at heavy rewriting, but because they regard equations as undirected, they often prove theorems that require the reverse orientation of a rewrite rule. Higher-order problems can be tackled, but the success rate is better for first-order problems. You may get better results if you first simplify the problem to remove higher-order features.

Note that problems can be easy for *auto* and difficult for ATPs, but the reverse is also true, so don't be discouraged if your first attempts fail. Because the system refers to all theorems known to Isabelle, it is particularly suitable when your goal has a short proof from lemmas that you don't know about.

Additional Commands

Several Isar commands are available to control the sledgehammer.

- `sledgehammer` $prover_1$ ... $prover_N$ invokes the specified automated theorem provers in parallel on the first subgoal. The first successful prover will terminate the others.
- The `print_atps` command tells about admissible prover names. Provers with the prefix `remote` query SystemOnTPTP, so an active internet connection is needed.
- If no provers are given as arguments to sledgehammer, the system refers to the default which is set to `"e spass remote_vampire"`.
- `atp_info` prints information about presently running provers.
- `atp_kill` terminates all running provers.

A sledgehammer for dependent type theory

Isabelle uses a foundational framework known as *simple type theory*.

Lean uses a foundational framework known as *dependent type theory*.

I have long argued that the latter is necessary for the kinds of algebraic reasoning that is essential to modern mathematics.

The greater distance from first-order logic, however, makes sledgehammer-type automation more complicated.

We are making progress, however.

A sledgehammer for dependent type theory

Our *LeanHammer* prototype uses the following:

- **Premise selection:** use a neural premise selector (Zhu, Clune, A, Jiang, Welleck) and an implementation of the Meng-Paulson symbolic heuristic by Kim Morrison
- **Translation:** use a monomorphization procedure, *Lean-auto* (Qian, Clune, Barrett, A)
- **External provers:** Zipperposition, cvc5
- **Reconstruction:** use *Duper* (Clune, Qian, Bentkamp, A) and *Aesop* (Limperg, From), or *Lean-SMT* (Mohamed, Mascarenhas, Khan, Barbosa, Reynolds, Qian, Tinelli, Barrett

Other automation for Lean

Josh Clune, Haniel Barbosa, and I are working on an alternative approach to proof reconstruction for cvc5:

- cvc5 reports theory-specific facts it used.
- These are proved by domain-specific automation like Grind and assembled by Duper.

Chase Norman is working on the Canonical prover:

- It's a complete search algorithm for dependent type theory.
- He has added monomorphization and domain-specific components.
- We are exploring the use of machine learning.

Small-scale automation in Lean

We use:

- **simp:** equational simplification
- **linarith:** linear real arithmetic
- **omega:** linear integer arithmetic
- **ring:** equational reasoning in rings
- **aesop:** a tableaux prover
- **norm-num:** numeric computation

These are all modeled after automation that has long been available in Isabelle and other systems.

A new Lean tactic, **grind**, combines and subsumes many of these.

Small-scale automation in Lean

Lean is its own metaprogramming language, making it possible for users to add their own automation.

Mathlib has:

- **mono:** for proving inequalities for using monotonicity properties
- **positivity:** for dispelling side conditions involving sign
- **gcongr:** for chaining inequalities with compound expressions
- **continuity:** for establishing continuity
- **fun_prop:** for proving continuity, measurability, differentiability, etc.
- **finiteness:** for proving $t \neq \infty$ in e.g. the extended reals.

Combination with machine learning

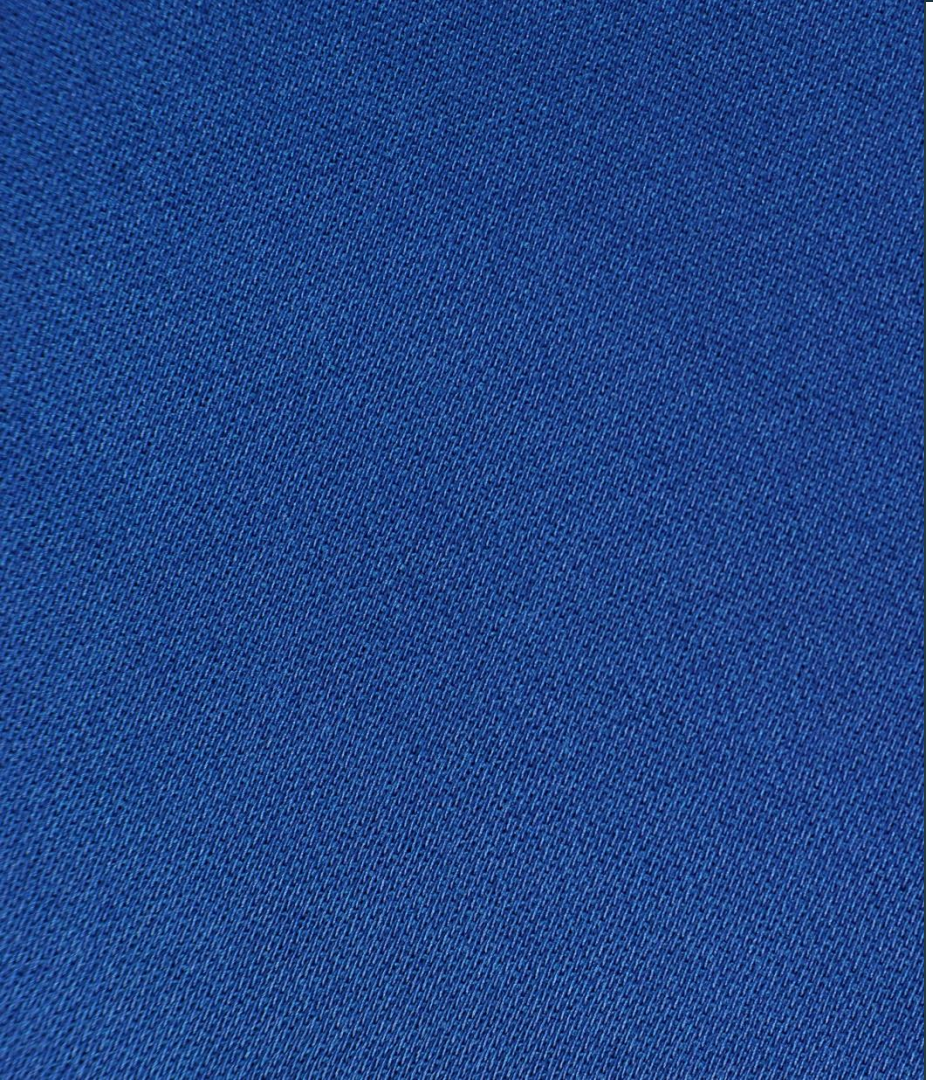
Currently, small-scale automation is used everywhere in Mathlib.

The sledgehammer and AI provers and copilots are getting better.

A sledgehammer is an example of neurosymbolic reasoning:

- a neural network finds relevant premises
- symbolic automation fills in small proofs

Other combinations are possible; see the next lecture.



Trying it out

Trying it out

- Take a look at Bernardo Subercaseaux's SAT For Mathematics web pages, with
  - a bibliography of applications of SAT solvers to mathematics
  - tutorials you can run in Colab notebooks.
- You can play the SAT game online.
- Take a look at the TPTP (Thousand of Problems of Theorem Provers) pages and SMTLib pages.
- You can find the Equational Theories Project and other projects discussed here online.
- The Logic and Mechanized Reasoning course will let you call automation from within Lean.

Conclusions

Conclusions

We have explored different uses of symbolic automation:

- discovering new mathematics
- verifying mathematics

The impact on mathematics have been minimal so far, but I think that will change.

Symbolic automation can be used synergistically with machine learning.

The tools are accessible.

We have a lot to learn.