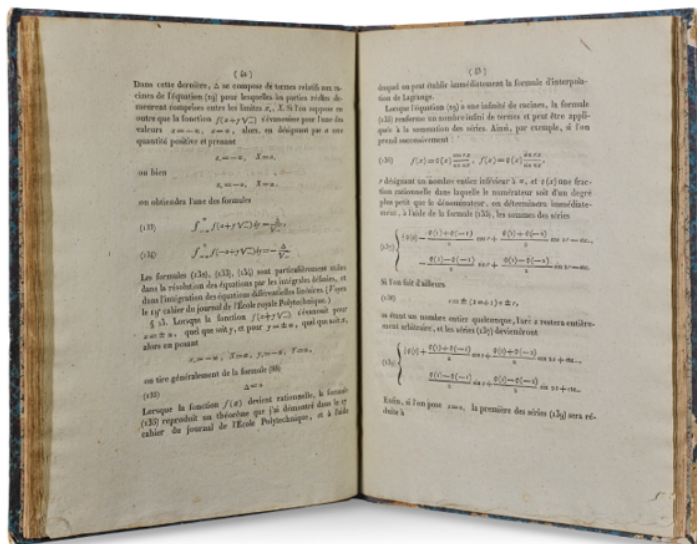


The Equational Theories Project: advancing collaborative mathematical research at scale

Matthew Bolan, Joachim Breitner, Jose Brox, Mario Carneiro, Martin Dvořák, Andrés Goens, Aaron Hill, Harald Husum, Zoltan Kocsis, Bruno Le Floch, Lorenzo Luccioli, Alex Meiburg, Pietro Monticone, Pace Neilsen, 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

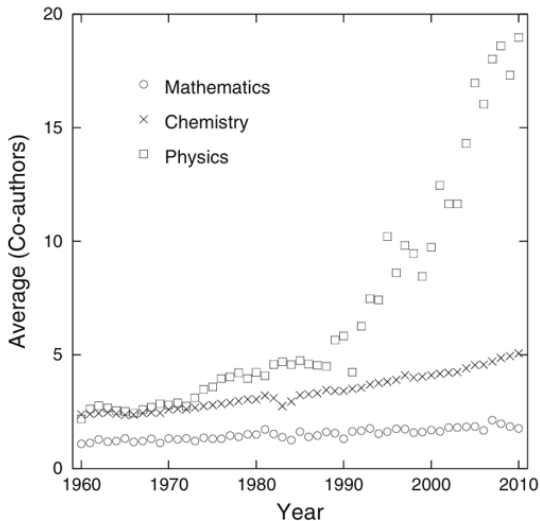
September 15, 2025

The way we conduct and disseminate mathematical research has not changed fundamentally in centuries.



(Mémoire sur les intégrales définies, Augustin-Louis Cauchy, 1825)

Due to high barriers of entry, our standard of 100% rigorous proof, and non-scaleable workflows, large-scale collaboration in mathematics is rare.



(Ding-wei Huang, Scientometrics, 2015)

But, with modern collaborative platforms, more usable formal proof verification tools, and promising experiments AI assistance, new research paradigms become possible:

- Large-scale mathematical research projects (“population surveys” rather than “case studies”)
- Broad participation (“citizen mathematics”)
- Data-driven mathematics
- Transparent dissemination of both process and results

Large-scale formalization projects

In recent years a “standard” recipe for a collaborative formalization project has emerged:

- Create a Github repository to host the formalized proofs (in a language such as Lean), together with a human-readable “blueprint” to break up complex theorems into more atomic lemmas that can be proven in a modular fashion.
- Coordinate in some openly accessible forum (e.g., the Lean Zulip chat, or via Github issue tracking) on how to assign tasks to contributors, and to make necessary adjustments to the initial blueprint.
- Allow for contributions from the general public, for instance via git Pull Requests (using Continuous Integration (CI) tools to verify their correctness).

Examples of such projects include

- The Liquid Tensor Experiment (2021–2022)
- Formalization of the Polynomial Freiman–Ruzsa conjecture (2023–2024+)
- Prime Number Theorem+ (2024–)
- Carleson’s theorem (2024–2025)
- Fermat’s Last Theorem (2024–)
- Exceptions to the abc conjecture (2025)

Some general lessons learned:

- A well-designed blueprint is key: ideally, pedantic details that would often be treated as “obvious” should be spelled out carefully, and proofs should be broken up into many small lemmas rather than into a few larger propositions.
- The structure should be modular, with the components of the proof self-contained except for a few very specific dependencies.
- Key definitions that are to be used throughout the project need to be extensively discussed and agreed upon early in the project (although refactoring a formal project can be easier than refactoring an informal one!).

The optimal writing style for a machine-checked proof is different from that of proof intended for human readers:

- Machine-checked analysis proofs are easier to write with explicit constants, whereas human-readable analysis proofs are easier using asymptotic notation that hides the constants.
- Human proofs of complex inequalities often proceed by a long linear sequence of simpler inequalities, but it can be easier in formalization to just state and prove all relevant inequalities and then call a tactic such as `linarith` to close.
- The deepest and most innovative portions of a proof can be surprisingly straightforward to formalize, whereas “obvious” steps can require non-trivial effort in the formalization. (The final result needed for PFR was the lemma that $\{1\}$, $\{2\}$, and $\{3, 4\}$ were pairwise disjoint.)

Collaborative research projects

But can this paradigm be used to create *new* mathematical theorems, rather than “just” formalizing ones that already exist as informal proofs?

Currently there are just a handful of projects in this regard:

- The Busy Beaver Challenge (BBC), 2022–2024
- The Equational Theories project (ETP), 2024–2025

The BBC was only formalized (in Coq) in a late stage of the project; but the ETP was designed to be formalized (in Lean) from the beginning.

Most of this talk will focus on describing the ETP.

ETP: origins

On July 16, 2023, the following question was asked on MathOverflow about the relation between two algebraic laws of a single binary operation.

Is there an identity between the commutative identity and the constant identity?

Asked 1 year, 10 months ago Modified 6 months ago Viewed 3k times



27



I asked this on Math Stack Exchange, but it didn't get a single answer. So, I am now asking it here. Let our signature be that of a single binary operation $+$. I define the constant identity to be $x + y = z + w$. The commutative identity is, of course, the well-known identity $x + y = y + x$. I wonder, is there an identity strictly between those two, meaning, is there a single identity E such that the constant identity implies E , but not conversely, and E implies the commutative identity, but not conversely?

universal-algebra

Edit tags

Share Cite Edit Follow Close Flag Protect

asked Jul 16, 2023 at 20:00



user107952

Featured on Meta

How Can We Improve Our Ecosystem? Community

Thoughts on the future of site customisation

2025 Community

2025 Community Results

Hot Meta Post

29 Minimal reputation potential mod

It was swiftly answered by another MathOverflow participant.



Yes:

38

$$(x + x) + y = y + x$$



The constant identity implies this because both sides are $+$ es. This does not imply the constant identity because it is true about any set with an operation that is commutative, associative, and idempotent (meaning $x + x = x$ for all x), the smallest nontrivial example is $\{x, y\}$ where $x + x = x + y = y + x = x$ and $y + y = y$.



This implies commutativity. $(x + x) + (x + x) = (x + x) + x = x + x$, so $x + x$ is idempotent. $(x + x) + y = ((x + x) + (x + x)) + y = y + (x + x)$, so $x + x$ commutes with everything. $(x + x) + (y + y) = (y + y) + x = x + y$, and $(x + x) + (y + y) = (y + y) + (x + x)$ because $x + x$ commutes with everything, so $x + y = y + x$, so $+$ is commutative.

This is not implied by commutativity, because (for example) addition of natural numbers is commutative but does not satisfy this identity.

Share Cite Edit Follow Flag

answered Jul 16, 2023 at 23:50



paste bee

The next day, a followup question was posted.

Is there an identity between the associative identity and the constant identity?

Asked 1 year, 10 months ago Modified 6 months ago Viewed 4k times



18



This is a follow-up to my previous question, here: [Is there an identity between the commutative identity and the constant identity?](#). Let our signature be that of a single binary operation $+$. I define the constant identity to be $x + y = z + w$. The associative identity is $(x + y) + z = x + (y + z)$. Is there an identity strictly between the two? Meaning, is there an identity E such that the constant identity implies E but not conversely, and E implies the associative identity but not conversely? Since the answer to my previous question was affirmative, I suspect the answer to this question will be affirmative also. But I can't find such an identity.

universal-algebra

Edit tags

Share Cite Edit Follow Close Flag

asked Jul 17, 2023 at 14:25



user107952

Featured on Meta

- How Can We Bring More I Ecosystem? Community Ic
- Thoughts on the future of site customisation
- 2025 Community Modera
- 2025 Community Modera Results

Hot Meta Posts

- 29 Minimal reputation requir potential moderators

I posted an answer (along with Pace Nielsen)



An identity E that obeys all the claimed properties is

22

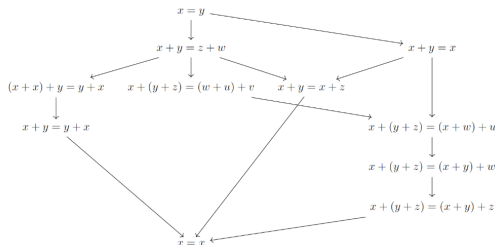
$$E : x + (y + z) = (x + y) + w \text{ for all } x, y, z, w.$$



- E is implied by triple constancy (and hence by constancy): obvious since both sides are constant in this case
- E does not imply triple constancy (and hence does not imply constancy either): follows from considering the [left-zero semigroups](#) $x + y = x$ mentioned by arsmath
- E implies associativity: obvious by specializing to $w = z$
- E is not implied by associativity: follows from considering (say) addition on the integers

This candidate E was located by pursuing the analysis in Pace's answer to isolate the form that E had to take as much as possible, as described in the comments to that answer. With a little more effort, it should be possible to entirely classify (up to relabelings and symmetries) the full set of identities E that answer the question.

Here is the [Hasse diagram](#) of the various identities discussed on this page and on the [related question linked by the OP](#), where the ordering is from stronger identities to weaker ones:



I thought it might be an interesting “graduate student project” to try to extend this graph further.

It might be a suitable undergraduate research project to extend this diagram to cover other short identities for magmas. EDIT: It might be a suitable graduate research project to find a way to do this automatically using proof assistants and possibly also machine learning/AI tools. (Some further

I proposed this on my social media. Some respondents found the idea interesting, but no-one volunteered to actually try this out.



Terence Tao

@tao

Jul 18, 2023, *

@ProfKinyon @caten Perhaps I can pose a concrete challenge to focus the discussion. Consider the collection of all possible universal equational laws in which the magma operation $+$ is applied at most four times. For instance, the associative law $(x+y)+z=x+(y+z)$ is of this form. I don't know exactly how many such laws exist, up to relabeling and symmetries, but let's say for sake of argument that there are around a thousand. The (admittedly artificial) challenge is to completely specify the partial ordering on this collection given by implication, thus extending the diagram included here to the other thousand or so identities involving at most four operations. Is this feasible to accomplish by some combination of current automated tools and reasonable number of human expert-hours? Note that one has not only to prove those implications that happen to be true, but also to disprove those implications that happen to be false. One could blindly throw random implications into various automated theorem provers, but would it actually be computationally feasible to completely specify the million or so implications this way? Or would some sort of automated "intuition" coming perhaps from machine learning tools be a significant speedup?

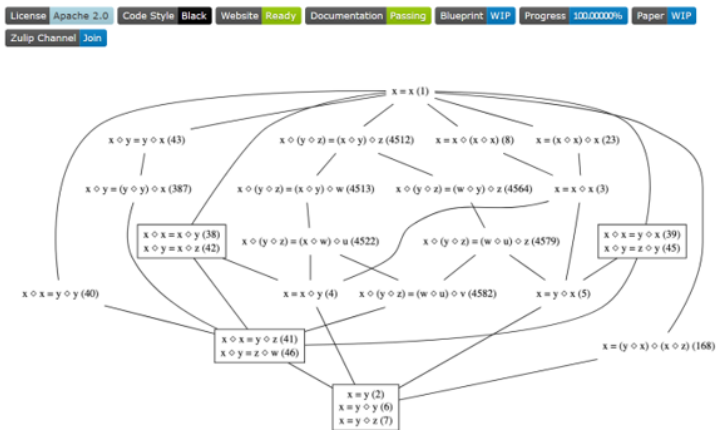
Over a year later, after concluding the successful PFR formalization, I was looking for a candidate pilot project to explore ways to use modern collaboration and formalization tools to conduct research.

From past experience, I expected the following features of such a project to be favorable:

- Modularity: the problem should be decomposable into many loosely connected pieces.
- Verifiability: correctness of contributions can be easily (or automatically) checked.
- Elementarity: a relatively low level of mathematical sophistication required to understand any component problem.
- Diversity of technique: problems should be susceptible to both human and computer-generated approaches of varying depth.
- Transferability: a solution to one component problem should help discover solutions to other component problems.
- Precisely defined goal, with a quantitative metric of progress: in particular, even small incremental contributions can be seen to “move the needle”.
- Visualizability: there should be some appealing way to

The problem I had proposed previously seemed to fit the bill, leading to the *Equational Theories Project*.

Equational theories project



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

What was the precise goal?

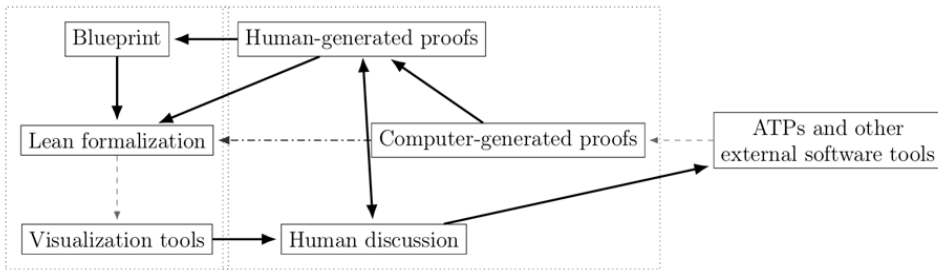
- We restricted attention to theories of a single equational law on a *magma*: a set M equipped with a binary operation $\diamond : M \times M \rightarrow M$.
- Examples include the *commutative law* $x \diamond y = y \diamond x$ (E43), the *associative law* $(x \diamond y) \diamond z = x \diamond (y \diamond z)$ (E4512), and the *singleton law* $x = y$ (E2).
- It turns out that up to trivial equivalences, there are 4694 laws that involve at most four applications of the magma operation; we selected a standardized numbering system to catalogue them.
- Define the *implication graph* between these laws by drawing an edge $En \vdash Em$ if the law En entails Em , i.e., every magma that obeys En necessarily obeys Em .

- For example, the associative law E4512 does not entail the commutative law E43, because there are associative magmas that are not commutative.
- On the other hand, the left absorptive law $x = x \diamond y$ (E4) entails the law $x = x \diamond (y \diamond x)$ (E10).
- The 4694 laws generated 22028942 potential edges, each of which was an implication $En \vdash Em$ or an anti-implication $En \not\vdash Em$.
- The project was to determine the graph by solving these 22028942 algebra problems.

- Many implications or anti-implications are easy to work out by hand - say with an hour on pen and paper. But this does not scale.
- On the other hand, knowledge of some portions of the graph can help fill in others. For instance, if one knows that En implies Em , and Em implies Ek , then En implies Ek .
- There is also a *duality symmetry* on the graph formed by replacing a magma operation $(x, y) \mapsto x \diamond y$ with its reflection $(x, y) \mapsto y \diamond x$.

- In general, the implication problem between two equational laws is known to be **undecidable**; counterexamples can require **infinite** magmas. Automated theorem provers (ATP)s can decide many specific implications, but are not guaranteed to succeed in finite time.
- A few specific laws are very well studied; for instance, Tarski's axiom $x = y \diamond (z \diamond (x \diamond (y \diamond z)))$ (E543) is known to precisely model subtraction on abelian groups. But most laws in the graph had no established literature.
- Resolving the 22028942 potential edges in the graph is beyond the capability of one human or computer tool; collaboration is needed.
- As *verifying* the 22028942 solutions is also beyond the capability of humans, formal verification is also needed.

After some trial and error, a viable workflow for the project was established, completing the graph within three months.



Individual proofs were proposed by humans...

Equational > 1648 \Rightarrow 206 ✓ ⌂ ⋮

OCT 14, 2024



Bernhard Reinke



7:46 AM

Hello, I think I have a counterexample that shows that 1648 does not imply 206. The magma is supported on the integers and defined as follows

$$x \diamond y = x - \text{sign}(y - x)$$

In other words, $x \diamond x = x$, and $x \diamond y = x + 1$ if $x > y$, and $x \diamond y = x - 1$ if $x < y$.

Now this satisfies 1648: it is clear if $x = y$, otherwise consider the case $x > y$, then

$$(x \diamond y) \diamond ((x \diamond y) \diamond y) = (x + 1) \diamond ((x + 1) \diamond y) = (x + 1) \diamond (x + 2) = x$$

the case $x < y$ is analogous. But this magma does not satisfy 206:

indeed, for $x = 0, y = -1$ we have

$$(x \diamond (x \diamond y)) \diamond y = (0 \diamond 1) \diamond -1 = -1 \diamond -1 = -1 \neq 0 = x$$



4

... or by various computer code runs.

Equational > Refutations using Z3



SEP 27,

Daniel Weber EDITED

😊 ⋮ ☆ 12:

Using Z3 I was able to refute $\text{Equation4283}[x \circ (x \circ y) = x \circ (y \circ x)] \Rightarrow \text{Equation4358}[x$

$\circ (y \circ z) = x \circ (z \circ y)]$:

x|0|1|2|3|4|5|6

---|---|---|---|---|---|---

0|1|1|1|4|1|6|1

1|1|1|3|1|1|1|1

2|1|5|1|1|5|1|5

3|1|1|1|1|1|1|1

4|1|1|3|1|1|1|1

5|1|1|1|1|1|1|1

6|1|1|3|1|1|1|1

With $x = 0$, $y = 1$, $z = 2$

The two modes were synergistic: a successful human proof of an implication could motivate another contributor to code up an automated search for other implications that could be settled by the same method...

OCT 14, 2024



Daniel Weber

😊 ⋮ ☆ 12:08 AM

Daniel Weber said:

Only the rules are important, right? Once we have them I think it should be immediate how R' should be defined, if we restrict it to only differ from R in $R'(a, b, c)$ and $R'(c, -, -)$, $R'(-, c, -)$ (and any values we have to set to satisfy old rules should be possible to simply convert to new rules)

This is false - in 1692 there's the rule $a \diamond a = x, b \diamond a = a, a \diamond b = c \rightarrow a \diamond x = c$, I need to think for a bit how to account for that

EDITED It succeeded on 118, 124, 476, 503, 677, 707, 713*, 883, 906, 1112, 1113, 1289*, 1447. The * are equations it seemingly succeeded on, but couldn't prove that all rules are preserved. I'll try to run it on the remaining equations with more time

1:17 AM



Terence Tao

9:11 AM

Thats a pretty good success rate!

When you are done with your run, can you update `Conjectures.lean` with all the implications that this procedure will likely resolve (presumably in the negative), so that it is clear to the other participants what remains unresolved even in advance of the Lean formalization? I imagine that

9:54 AM

...or a computer-generated proof would be “deconstructed” to glean human-comprehensible insights.

Equational > FINITE: The Lean+Duper implications ✓ ·)) :

DEC 5,

Amir Livne Bar-on EDITED

😊 ⋮ ☆ 2:3

Looking at the Duper proofs of $1167 \Rightarrow X$ in

[https://leanprover.zulipchat.com/#narrow/channel/458659-](https://leanprover.zulipchat.com/#narrow/channel/458659-Equational/topic/Austin.20pairs/near/481257624)

[Equational/topic/Austin.20pairs/near/481257624](https://leanprover.zulipchat.com/#narrow/channel/458659-Equational/topic/Austin.20pairs/near/481257624), many of them go through 4689 (spelled "eq2693" in DuperConjectures1.lean). This was transitively reduced to the conjecture $1167 \Rightarrow 4615$.

Here's a proof for that one:

Equation 1167 is $L_y L_{z \diamond S y} = I$, while equation 4615 is $L_{S x} = L_{z \diamond x}$. For x s that are squares $x = S a$ we have $L_{z \diamond x} = L_a^{-1}$, which is independent of z so 4615 holds.

But every element in 1167 is a square: $S x = L_x x$, so $x = L_{z \diamond S x} S x$ for any z . We can take for example $a = L_{S x \diamond S x} S x$ for the above.

Formalization of these results into Lean was accomplished by hand...

Equational > 1648 \Rightarrow 206 ✓ ⌂ ⋮

OCT 14, 2024



Shreyas Srinivas EDITED

8:40 AM

Here's a long flow-of-thought proof. I think this can be golfed down significantly if `split` allowed naming the hypothesis and I used more mathlib defs:

```
import Mathlib.Tactic

class Magma (α : Type _) where
  op : α → α → α

infix:65 " ⋄ " => Magma.op

abbrev Equation1648 (G: Type _) [Magma G] := ∀ x y : G, x = (x ⋄ y) ⋄ ((x ⋄ y) ⋄ y)

abbrev Equation206 (G: Type _) [Magma G] := ∀ x y : G, x = (x ⋄ (x ⋄ y)) ⋄ y

def sign (x : ℤ) :=
  if x = 0 then 0 else if x < 0 then -1 else 1

theorem Equation1648_not_implies_Equation206 : ∃ (G: Type) (_: Magma G), Equation1648 G
  let instMagmaInt : Magma ℤ := {
    op := fun x y => x - sign (y - x)
  }
  use ℤ, instMagmaInt
  simp [Equation1648, Equation206]
  constructor
```

...or by further *ad hoc* code.



Stefan Hetzl

1:54 PM

I wrote some code that takes an implication between two equations, 1. sends it to an automated theorem prover, e.g., prover9, and 2. translates the returned proof into Lean code to prove the original implication.

This work is based on the [gapt system](#). You can find the code in the [eqthproject branch](#) at [examples/eqthproject/eqthproject.scala](#).

The import from various automated theorem provers into gapt exists since a long time and is quite stable. The system also provides a translation from resolution proofs to sequent calculus proofs. What I implemented now is a rather naive (line-by-line) translation of simple sequent calculus proofs (such as those of the implication of one equation by another) to Lean code.

For example, it automatically generates code like



```
theorem eqimpl (G: Type*) [Magma G] (h0: ∀ (x y :G), ( x • x ) = ( x • y )): ( ∀
  intro a1 a2 a3
  have h1: ( a1 • a1 ) = ( a1 • a1 ) -- cut
  · rfl -- refl
  · have h2 := h0 -- c:l
    have h3: ( a1 • a3 ) = ( a1 • a1 ) -- cut
    · have h4: ( a1 • a1 ) = ( a1 • a3 ) -- cut
      · have h5: ∀ y, ( a1 • a1 ) = ( a1 • y ) := by apply h2 -- ∀:l
```


The project was rather elementary in nature and only had a modest reliance on Lean's Mathlib. Nevertheless a few technical decisions on the Lean framework had to be made at the start of the project, such as

- Avoiding the use of Lean extensions such as `egg` or `duper` in the base repository, although they were used to generate “unofficial” proofs
- Using a custom `Magma` instance rather than Mathlib's `Mul` or `Add`
- Implementing equational laws both semantically (as a first-order sentence on magmas) and syntactically (as a pair of elements of a `FreeMagma`)

Many tasks were assigned as Github issues, for any volunteer to take on. In particular, the formalization of a proof could be performed by a different contributor than the one who initially provided the proof.


FINITE: Establish 1441->4067, 1443->3055, 1681->3877, 1701->1035 #986



 Closed  #1015






teorth opened on Dec 5, 2024 · edited by teorth Edits ▾ ⋮


Turn the conjectures to theorems in
https://github.com/teorth/equational_theories/blob/main/equational_theories/ManuallyProved/Equation1441.lean
using the proofs provided in the blueprint
When done, add `\lean` and `\leanok` tags to the blueprint as appropriate.

Create sub-issue ▾ 

 teorth added this to  [Equational Theories Project](#) on Dec 5, 2024


 teorth converted this from a draft issue on Dec 5, 2024

 teorth mentioned this on Dec 5, 2024
 FINITE: Convert the Lean+Duper proofs from 467, 1133, 1167, 1441, 1443, 1681, 1701 to Lean proofs #935



vlad902 on Dec 10, 2024 Contributor ⋮

claim



A few implications were particularly intricate and required one first create a human-readable *blueprint* intermediate between the original human-generated proof, and the final Lean formalization.

Now we seek to enlarge a partial solution. We first make an easy observation:

Proposition 18.13. (Enlarging L'_0)

Suppose one has a partial solution in which L'_0x is undefined for some $x \in N$. Then one can extend the partial solution so that L'_0x is now defined.

Proof ▼

By axiom (i''), $L'_0(R'_0)^n x$ is undefined for every integer n . Let $d = E_m$ be a generator of SM that does not appear as a component of any index of any of the generators e_a appearing anywhere in the partial solution; such a d exists due to the finiteness hypotheses. We set $L'_0x := e_d$, and then extend by [Equation 6](#), [Equation 7](#), thus

$$L'_0(R'_0)^n x := (R'_0)^n e_d$$

and

$$L'_0(R'_0)^n e_d := (R'_0)^{n-1} x.$$

Because of the new nature of d , no collisions in the partial function L_0 are created by

Tools were developed to visualize progress and identify key potential implications to focus on. This facilitated independent explorations of the graph by many different contributors, with almost no centralized guidance.



Dashboards listing the current number of implications and anti-implications resolved were a convenient way to measure progress.

Day 16 (Oct 11)

Explicitly true	Implicitly true	Explicitly false	Implicitly false	No proof
10657	8167622	582156	13272313	888

Of the unproven claims, we conjecturally have

Explicitly true	Implicitly true	Explicitly false	Implicitly false	No conjecture
0	0	174	10	704

Most of the implications were “low-hanging fruit” that could be resolved by relatively simple techniques, such as:

- Brute force use of ATPs
- Brute force testing of small finite magmas (e.g., all magmas of order at most 4)
- Testing of special magmas, such as linear models $x \diamond y = ax + by + c$ on a ring, or translation-invariant models $x \diamond y = x + f(y - x)$
- Applying transitive closure or duality

This reduced the original set of 22028942 problems to about a thousand.

This initial filtering allowed key model problems to come into view, which stimulated the development of new techniques. For instance: did the “Asterix” equation $x = y \diamond (x \diamond (y \diamond x))$ (E65) imply the “Obelix” equation $x = (y \diamond x) \diamond (y \diamond (y \diamond x))$ (E1491)?

- For finite magmas, we could show the answer was yes (because injective maps $f : M \rightarrow M$ were necessarily invertible for finite M).
- But for infinite magmas, we developed a **greedy construction** method to show that the answer was no.
- The same method could be automated to settle hundreds of other anti-implications.

Several other techniques were also discovered in this fashion, including

- Human-guided use of ATPs to generate large finite counterexamples
- “Magma cohomology” techniques to construct useful “extensions” of a base magma
- Discovery of a “twisting semi-group” invariant attached to each law
- Automated ways to determine if a law was *confluent*, or admitted a *complete rewriting system*

A followup project was launched to determine the *finite implication graph* - to determine which implications $En \vdash_{\text{fin}} Em$ held for *finite* magmas M . Here, all 22028942 implications were resolved except for one (and its dual):

Open problem

Does the law $x = y \diamond (x \diamond ((y \diamond x) \diamond y))$ (E677) imply the law $x = ((x \diamond x) \diamond x) \diamond x$ (E255) for finite magmas?

Several partial results on this problem are collected in the blueprint for the project.

The dog that did not bark in the night - modern ML/AI

- The ETP made extensive, crucial use of “good old-fashioned AI” (GOFAI), mostly in the form of **automated theorem provers**.
- Surprisingly, though, modern machine learning and AI tools only had modest uses: building useful visualization tools, some code completion, and guessing a rewriting system for a specific law based on other examples.
- A posteriori, it appears that the implication graph has some structure, in that a neural network can reconstruct it with high accuracy given a significant subportion of the graph. But we did not utilize such tools during the main phase of the project.

Future directions

- This project was chosen to have a maximally favorable environment for collaborative research.
- One potential bottleneck for similar projects would be the difficulty in formalizing any incremental contribution into Lean, if the topic is not as elementary as the algebra of magmas.
- For instance, a project in analysis that required synthesizing thousands of simple estimates could be tricky to implement in Lean due to the limited support for asymptotic estimates.
- Writing such contributions in a lightweight (and not fully verified) proof assistant may be needed as an intermediate step to complete formalization.

Thanks for listening!

