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 sought to determine the implication graph between 4692 equational laws on magmas, by a combination of human-generated and automated proofs, all validated by the formal proof assistant language *Lean*. **state key outcomes**

The Equational Theories Project

Matthew Bolan, Mario Carneiro, Alex Meiburg, Pietro Monticone, Terence Tao, Vlad T

November 22, 2024

Contents

1 Introduction **4**

1.1 Magmas and Equational Laws 4

1.2 The Equational Theories Project 6

1.3 Outcomes 7

2 Notation and Mathematical Foundations **8**

3 Formal Foundations **9**

4 Project Management **10**

4.1 Handling Scaling Issues 11

4.2 Other Design Considerations 11

4.3 Maintenance 12

5 Counterexample constructions **12**

5.1 Finite magmas 13

5.2 Linear models 13

5.3 Translation-invariant models 13

5.4 Ad hoc constructions 13
5.5 Greedy constructions 14
5.6 Modifying base models 14

6 Syntactic arguments **14**

6.1 Simple rewrites 14
6.2 Invariants 14
6.3 Confluence 14
6.4 Complete rewriting systems 14
6.5 Unique factorization 14

7 Automated Theorem Proving **14**

8 Implications for Finite Magmas **15**

9 AI and Machine Learning Contributions **15**

10 User Interface **16**

11 Statistics and Experiments **16**

12 Data Management **17**

13 Reflections **17**

14 Conclusions and Future Directions **17**

A Numbering system **18**

B Proofs of Theoretical Results **19**

1 Introduction

The purpose of this paper is to report on the *Equational Theories Project* (ETP)[1], a pilot project launched[2] in September 2024 to explore new ways to collaboratively work on mathematical research projects using machine assistance. The project goal, in the area of universal algebra, was selected to be particularly amenable to crowdsourced and computer-assisted techniques, while still being of mathematical research interest. **Describe outcomes**

1.1 Magmas and Equational Laws

In order to describe the mathematical goals of the ETP, we need some notation. A *magma* $M = (M, \diamond)$ is a set M (known as the *carrier*) together with a binary operation $\diamond\colon M \times M \to M$. An *equational law* for a magma, or *law* for short, is an identity involving $\diamond$ and some indeterminates, which we will typically denote using the symbols x, y, z, u, v, w. Familiar examples of equational laws include the *commutative law*

$$x \diamond y = y \diamond x \tag{E43}$$

and the *associative law*

$$(x \diamond y) \diamond z = x \diamond (y \diamond z). \tag{E4512}$$

In the ETP, a unique number was assigned to each equational law, via a numbering system that we describe in Appendix A. Formally, one can represent an equational law syntactically as a string $w_1 \simeq w_2$, where w_1, w_2 are words in a free magma generated by formal indeterminate symbols; see ???.

A magma M obeys a law E if the law E holds for all possible assignments of the indeterminate to M, in which case we write $M \models E$. Thus for instance $M \models E43$ if one has $x \diamond y = y \diamond x$ for all $x, y \in M$.

We place a pre-order on laws by writing $E \leq E'$ or $E \vdash E'$ if every magma that obeys E, also implies E': $(M \models E) \implies (M \models E')$. We say that two laws are *equivalent* if they imply each other. For instance, the constant law

$$x \diamond y = z \diamond w \tag{E46}$$

can easily be seen to be equivalent to the law

$$x \diamond x = y \diamond z. \tag{E41}$$

[1] `https://teorth.github.io/equational_theories/`

[2] `https://terrytao.wordpress.com/2024/09/25`

In this pre-ordering, a maximal element is given by the trivial law

$$x = x \tag{E1}$$

and a minimal element is given by the singleton law

$$x = y. \tag{E2}$$

The *order* of an equational law is the number of occurrences of the magma operation. For instance, the commutative law (E43) has order 2, while the associative law (E4512) has order 4. We note some selected laws of small order that have previously appeared in the literature:

- The *central groupoid law*
$$x = (y \diamond x) \diamond (x \diamond z) \tag{E168}$$
is an order 3 law introduced by Evans [2] and studied further by Knuth [6] and many further authors, being closely related to central digraphs (also known as unique path property diagraphs), and leading in particular to the discovery of the Knuth-Bendix algorithm [7]; see [8] for a more recent survey.

- *Tarski's axiom*
$$x = y \diamond ((z \diamond (x \diamond (y \diamond z)))) \tag{E543}$$
is an order 4 law that was shown by Tarski [16] to characterize the operation of subtraction in an abelian group; that is to say, a magma M obeys (E543) if and only if there is an abelian group structure on M for which $x \diamond y = x - y$ for all $x, y \in M$.

- In a similar vein, it was shown in [13] (see also [14]) that the order 4 law
$$x = (y \diamond z) \diamond (y \diamond (x \diamond z)) \tag{E1571}$$
characterizes addition (or subtraction) in an abelian group of exponent 2; it was shown in [11] that the order 4 law
$$x = (y \diamond ((x \diamond y) \diamond y)) \diamond (x \diamond (z \diamond y)) \tag{E345169}$$
characterizes the Sheffer stroke in a boolean algebra, and it was shown in [4] that the order 8 law
$$x = y \diamond ((((y \diamond y) \diamond x) \diamond z) \diamond (((y \diamond y) \diamond y) \diamond z)) \tag{E42323216}$$
characterizes division in a (not necessarily abelian) group.

Some further examples of laws characterizing well-known algebraic structures are listed in [10].

The Birkhoff completeness theorem [1, Th. 3.5.14] implies that an implication $E \vdash E'$ of equational laws holds if and only if the left-hand side of E' can be transformed into the

right-hand side by a finite number of substitution rewrites using the law E. However, the problem of determining whether such an implication holds is undecidable in general [12]. Even when the order is small, some implications[3] can require lengthy computer-assisted proofs; for instance, it was noted in [5] that the order 4 law

$$x = (y \diamond x) \diamond ((x \diamond z) \diamond z) \tag{E1689}$$

was equivalent to the singleton law (E2), but all known proofs are computer-assisted.

1.2 The Equational Theories Project

As noted in Appendix A, there are 4694 equational laws of order at most 4. The primary mathematical goal of the ETP was to completely determine the implication pre-ordering $\leq$ for this set of laws. Such systematic determinations of implication graphs have been seen previously in the literature; for instance, in [15], the relations between 60 identities of Bol-Moufang type were established, and in the blog post [17, §17], some initial steps towards generating this graph for the first hundred or so laws on our list were performed. However, to our knowledge, the ETP is the first project to study such implications at the scale of thousands of laws.

The ETP requires the determination of the truth or falsity of $4694^2 = 22033636$ implications; while one can use properties such as the transitivity of the pre-ordering to reduce the work somewhat, this is clearly a task that requires significant automation. It was also a project highly amenable to crowdsourcing, in which different participants could work on developing different techniques, each of which could be used to fill out a different part of the implication graph. In this respect, the project could be compared with a Polymath project [3], which used online forums such as blogs and wikis to openly collaborate on a mathematical research problem. However, the Polymath model required human moderators to review and integrate the contributions of the participants, which clearly would not scale to the ETP which required the verification of over twenty million mathematical statements. Instead, the ETP was centered around a Github repository in which the formal mathematical contributions had to be entered in the proof assistant language *Lean*, where they could be automatically verified. In this respect, the ETP was more similar to the recently concluded Busy Beaver Challenge[4], which was a similarly crowdsourced project that computed the fifth Busy Beaver number $BB(5)$ to be 47176870 through an analysis of 88664064 Turing machines, with the halting analysis being verified in a variety of computer languages, with the final formal proof written in the proof assistant language *Coq*. One of the aims of the ETP was to explore potential workflows for such collaborative, formally verified mathematical research projects that could serve as a model for future projects of this nature.

Secondary aims of the ETP included the possibility of discovering unusually interesting equational laws, or new experimental observations about such laws, that had not previously

[3] Another contemporaneous example of this phenomenon was the solution of the Robbins problem [9].

[4] `https://bbchallenge.org/`

been noticed in the literature; and to develop benchmarks to assess the performance of automated theorem provers and other AI tools.

1.3 Outcomes

The ETP achieved its primary objective, with all of the implications formalized in the proof assistant language *Lean*, and can be found on the ETP GitHub repository. The experience of running such a large collaborative research project introduced several challenges, which we report upon in Section 4. Also, a variety of methods with varying degrees of automation or computer-assistance had to be developed to resolve all the implications, which had quite a variety of difficulty levels.

Of the 22033636 possible implications $E \vdash E'$, 8178279 (or 37.12%) would end up being true. To establish such positive implications $E \vdash E'$, the main techniques used were as follows:

- A very small number of positive implications were established and formalized by hand, mostly through direct rewriting of the laws; but this approach would not scale to the full project.
- Simple rewriting rules, for instance based on the observation that any law of the form $x = f(y, z, \dots)$ was necessarily equivalent to the trivial law (E2), could already reduce the size of potential equivalence classes by a significant fraction. We discuss this method in Section 6.1.
- The preorder axioms for $\vdash$, as well as the "duality" symmetry of the preorder with respect to replacing a magma operation $x \diamond y$ with its opposite $x \diamond^{\mathrm{op}} y := y \diamond x$, can be used to significantly cut down on the number of implications that need to be proven explicitly; ultimately, only 10657 (0.05%) of the positive implications needed a direct proof.
- Automated Theorem Provers (ATP) could be deployed at acceptable compute cost to establish this generating set of positive implications, as discussed in Section 7.

More challenging were the 13855357 (62.88%) implications that were false, $E \nvdash E'$. Here, the range of techniques needed to refute such implications were quite varied.

- Syntactic methods, such as observing an "invariant" of the law E that was not shared by the law E', could be used to obtain some refutations. For instance, if both sides of E had the same order, but both sides of E' did not, this could be used to syntactically refute $E \vdash E'$. Similarly, if the law E was confluent, enjoyed a complete rewriting system, or otherwise permitted some understanding of the free magma associated to that law, one could decide the assertions $E \vdash E'$ for all possible laws E', or at least a significant fraction of such laws. We discuss these methods, and the extent to which

they can be automated in Section 6. One novel such invariant we introduce here is the "twisting semigroup" of an equational theory, which gave simple refutations of some otherwise very challenging implications.

- Small finite magmas, which can be described explicitly by multiplication tables, could be tested by brute force computations to provide a large number of counterexamples to implications, or by ATP-assisted methods. See Section 5.1.

- Linear models, in which the magma operation took the form $x \diamond y = ax + by$ for some (commuting or non-commuting) coefficients a, b, allowed for another large class of counterexamples to implications, which could be automatically scanned for either by brute force or by Grobner basis type calculations. See Section 5.2.

- Translation invariant models, in which the magma operation took the form $x \diamond y = x + f(y - x)$ on an additive group, or $x \diamond y = xf(x^{-1}y)$ on a non-commutative group, reduce matters to analyzing certain functional equations; see Section 5.3.

- Greedy methods, in which either the multiplication table $(x, y) \mapsto x \diamond y$ or the function f determining a translation-invariant model are iteratively constructed by a greedy algorithm subject to a well-chosen ruleset, were effective in resolving many implications not easily disposed of by preceding methods. See Section 5.5.

- Starting with a simple base magma M obeying both E and E', and either enlarging it to a larger magma $M' \supset M$, extending it to a magma N with a projection homomorphism $\pi : N \to M$, or modifying the multiplication table on a small number of values, also proved effective when combined with greedy methods. See Section 5.6.

- Some *ad hoc* models based on existing mathematical objects, such as infinite trees, rings of polynomials, or "Kisielewicz models" utilizing the prime factorization of the natural numbers, could also handle some otherwise difficult cases. In some cases, the magma law induced some relevant and familiar structures, such as a directed graph or a partial order, which also helped guide counterexample constructions. See Section 5.4.

- Automated theorem provers were helpful in identifying which simplifying axioms could be added to the magma without jeopardizing the ability to refute the desired implication $E \vdash E'$.

Mention other outcomes, e.g. finite magma graph

2 Notation and Mathematical Foundations

If M is a magma, we define the left and right multiplication operators $L_a, R_a \colon M \to M$ for $a \in M$ by the formula

$$L_x y = R_y x \coloneqq x \diamond y. \tag{1}$$

We also define the squaring operator $S\colon M \to M$ by

$$Sx \coloneqq x \diamond x = L_x x = R_x x \tag{2}$$

and the (right) cubing operator $C\colon M \to M$ by

$$Cx \coloneqq Sx \diamond x = R_x^2 x. \tag{3}$$

If X is an alphabet, we let M_X denote the free magma generated by X, thus an element of M_X is either a letter in X, or of the form $w_1 \diamond w_2$ with $w_1, w_2 \in M_X$. Every function $f\colon X \to M$ into a magma M extends to a unique homomorphism $\varphi_f\colon M_X \to M$. Formally, an equational law with some indeterminates in X can be written as $w_1 \simeq w_2$ for some $w_1, w_2 \in X$; a magma M then obeys this law if and only if $\varphi_f(w_1) = \varphi_f(w_2)$ for all $f\colon X \to M$.

Every magma M has an opposite M^{op}, which has the same carrier but the opposite operation $x \diamond^{\mathrm{op}} y \coloneqq y \diamond x$. A magma M obeys an equational law E if and only if its opposite M^{op} obeys the dual law E^*, defined by reversing the all operations. For instance, the dual of Equation 327,

$$x \diamond y = x \diamond (y \diamond z), \tag{E327}$$

is the the law $y \diamond x = (z \diamond y) \diamond x$, which in our numbering system we rewrite as Equation 395,

$$x \diamond y = (z \diamond x) \diamond y. \tag{E395}$$

We then see that the implication graph has a duality symmetry: given two equational laws E_1, E_2, we have $E_1 \vdash E_2$ if and only if $E_1^* \vdash E_2^*$.

3 Formal Foundations

TODO: expand this sketch.

Here we describe the Lean framework used to formalize the project, covering technical issues such as:

- Magma operation symbol issues
- Syntax (‘LawX‘) versus semantics (‘EquationX‘)
- ”Universe hell” issues
- Additional verification (axiom checking, Leanchecker, etc.)
- Use of the ‘conjecture‘ keyword

- Use of namespaces to avoid collisions between contributions. (Note: we messed up with this with FreeMagma! Should have namespaced back end results as well as front end ones.)
- Use of Facts command to efficiently handle large numbers of anti-implications at once

Upstream contributions:

- Mathlib contributions
- LeanBlueprint contributions

4 Project Management

TODO: expand this sketch.

Shreyas Srinivas and Pietro Monticone have volunteered to take the lead on this section.

Discuss topics such as:

- Project generation from template
- Github issue management with labels and task management dashboard
- Continuous integration (builds, blueprint compilation, task status transition)
- Pre-push git hooks
- Use of blueprint (small note, see #406: blueprint chapters should be given names for stable URLs)
- Use of Lean Zulip (e.g. use of polls)

Maybe give some usage statistics, e.g. drawing from `https://github.com/teorth/equational_theories/actions/metrics/usage`

Mention that FLT is also using a similar workflow.

4.1 Handling Scaling Issues

Also mention some early human-managed efforts ("outstanding tasks", manually generated Hasse diagram, etc.) which suffices for the first one or two days of the project but rapidly became unable to handle the scale of the project.

Mention that some forethought in setting up a Github organizational structure with explicit admin roles etc. may have had some advantages if we had done so in the planning stages of the project, but it was workable without this structure (the main issue is that a single person - Terry - had to be the one to perform various technical admin actions).

Use of transitive reduction etc. to keep the Lean codebase manageable. Note that the project is large enough that one cannot simply accept arbitrary amounts of Lean code into the codebase, as this could make compilation times explode. Also note somewhere that transitive completion can be viewed as directed graph completion on a doubled graph consisting of laws and their formal negations.

Technical debt issues, e.g., complications stemming from an early decision to make Equations.lean and AllEquations.lean the ground truth of equations for other analysis and visualization tools, leading to the need to refactor when AllEquations.lean had to be split up for performance reasons.

Note that the "blueprint" that is now standard for guiding proof formalization projects is a bit too slow to keep up with this sort of project that is oriented instead about proving new results. Often new results are stated and formalized without passing through the blueprint, which is then either updated after the fact, or not at all. So the blueprint is more of an optional auxiliary guiding tool than an essential component of the workflow.

4.2 Other Design Considerations

Explain what "trusting" Lean really means in a large project. Highlight the kind of human issues that can interfere with this and how use of tools like external checkers and PR reviews by people maintaining the projects still matters. Provide guidelines on good practices (such as branch protection or watching out for spurious modifications in PRs, for example to the CI). Highlight the importance of following a proper process for discussing and accepting new tasks, avoiding overlaps etc. These issues are less likely to arise in projects with one clearly defined decision maker as in this case, and more likely to arise when the decision making has to be delegated to many maintainers.

Note that despite the guarantees provided by Lean, non-Lean components still contained bugs. For instance, an off-by-one error in an ATP run created a large number of spurious conjectures, and some early implementations of duality reductions (external to Lean) were similarly buggy. "Unit tests", e.g., checking conjectured outputs against Lean-validated outputs, or by theoretical results such as duality symmetry, were helpful, and the Equation

Explorer visualization tool also helped human collaborators detect bugs.

Meta: documenting thoughts for the future record is quite valuable. It would be extremely tedious to try to reconstruct real-time impressions long after the fact just from the github commit history and Zulip chat archive.

4.3 Maintenance

Describe the role of maintainers and explain why they need to be conversant in the mathematics being formalised, as well as Lean. As such, the role of maintainers is often akin to postdocs or assistant profs in a research group who do some research of their own, but spend much of their time to guide others in performing their tasks, the key difference being that contributors are volunteers who choose their own tasks. Explain the tasks maintainers must perform. Examples:

- Reviewing proofs,
- Helping with proofs and theorem statements when people get stuck,
- Offering suggestions and guidance on how to produce shorter or more elegant proofs,
- Ensuring some basic standards are met in proof blocks that make proofs robust to upstream changes,
- Creating and maintaining CI processes,
- Responding to task requests,
- Evaluating theorem and definition formulations (for example unifying many theorem statements into one using FactsSyntax),
- Suggesting better ones where possible,
- Ensuring that there is no excessive and pointless overlap of content in different contributions (TODO: elaborate on what level of overlap was permissible and what we consider excessive).

5 Counterexample constructions

TODO: Expand this sketch

5.1 Finite magmas

Discuss semi-automated creation of finite counterexamples (as discussed here) Describe various sources of example magmas used in counterexamples, including the ones listed here.

Also note some "negative results" - classes of finite magmas that did not yield many additional refutations, e.g. commutative 5x5 magmas.

Mention finite immunity

Using SAT solvers to find medium sized finite magmas obeying a given law? See this discussion.

Discuss computational and memory efficiencies needed to brute force over extremely large sets of magmas. SAT solving may be a better approach past a certain size!

5.2 Linear models

Note both commutative and noncommutative models

Mention Lefschetz principle - commutative models can be made finite

Note that linear magmas let one assign an "affine scheme" to each law that can be used to rule out many, but not all, implications.

Mention Linear immunity

5.3 Translation-invariant models

Translation-invariant magmas (see e.g., this thread for a nice example). Note: any magma with a transitive symmetry will lift to a translation-invariant model, so this helps explain why these are common examples. Also symmetric models could be slightly more likely to obey various laws than general models due to degree of freedom considerations.

Mention translation-invariant immunity

5.4 Ad hoc constructions

Tree based constructions, see here.

5.5 Greedy constructions

5.6 Modifying base models

6 Syntactic arguments

TODO: Expand this sketch

6.1 Simple rewrites

6.2 Invariants

Mention free magmas as one source of invariants

6.3 Confluence

6.4 Complete rewriting systems

6.5 Unique factorization

Discuss the 854 example

7 Automated Theorem Proving

TODO: expand this sketch

Describe various automated theorem provers deployed in the project, with some statistics on performance:

- Z3 prover
- Vampire
- More elementary substitutions
- egg

- duper

Any comparative study of semi-automated methods with fully automated ones? In principle, the semi-automated approach could be more automated using a script or "agent" to call various theorem provers. See this discussion.

Draw upon the discussion here on the various ways of integrating ATPs with Lean. This project primarily used the least integrated approach, "Option 3", as it was fastest and required no dependencies on the other contributors, but it also has drawbacks.

Note: when evaluating the performance of any particular automated implication tool, we should do a fresh run on the entire base of implications, rather than rely on whatever implications produced by the tool remain in the Lean codebase by the time of writing the paper. This is because (a) many of the previous runs focused only on those implications that were not already ruled out by earlier contributions, and (b) some pruning has been applied subsequent to the initial runs to improve compilation performance. Of course, these runs would not need to be added to the (presumably optimized) codebase at that point, but would be purely for the purpose of gathering performance statistics. More discussion here.

See this discussion on the value of using different ATPs and setting run time parameters etc. at different values.

Make a note of the possible alternate strategy to prove implications outlined here.

8 Implications for Finite Magmas

TODO: Expand this sketch

Introduce Austin pairs.

Recap discussion from `https://leanprover.zulipchat.com/#narrow/channel/458659-Equational/topic/Austin.20pairs`

9 AI and Machine Learning Contributions

TODO: expand this sketch

- Claude assistance in coding front ends.
- ChatGPT to guess a complete rewriting system.

- Minor use of GitHub Copilot to autocomplete code in Lean and other languages.
- See this discussion.
- Directed Link Prediction (DLP) on the implication graph with multiple GNN autoencoders (see related Zulip topic at `https://leanprover.zulipchat.com/#narrow/channel/458659-Equational/topic/Graph.20ML.3A.20Directed.20link.20prediction.20on.20the.20implication.20graph`).

ML experiments to learn the implication graph.

10 User Interface

Describe visualizations and explorer tools: Equation Explorer, Finite Magma Explorer, Graphiti, ...

11 Statistics and Experiments

TODO: Expand this sketch

Data analysis of the implication graph:

- Mention the long chain $2 \Rightarrow 5 \Rightarrow 2499 \Rightarrow 2415 \Rightarrow 238 \Rightarrow 2716 \Rightarrow 28 \Rightarrow 2973 \Rightarrow 270 \Rightarrow 3 \Rightarrow 3715 \Rightarrow 375 \Rightarrow 359 \Rightarrow 4065 \Rightarrow 1$ (discussed here).
- What are the "most difficult" implications?
- Is there a way to generate a standard test set of implication problems of various difficulty levels? Can one then use this to benchmark various automated and semi-automated methods? Challenge: how does one automatically assign a difficulty level to a given (anti-)implication?

See this for a preliminary data analysis of the impact of equation size and the number of variables.

Analyze the implication graph and discuss test sets of implication problems for benchmarking theorem provers. Challenge: How can one automatically assign a difficulty level to an implication?

12 Data Management

TODO: expand this sketch

Describe how data was handled during the project and how it will be managed going forward.

13 Reflections

TODO: expand this sketch

Testimonies from individual participants (but perhaps this is more suited for a retrospective paper). Utilize the thoughts and reflections thread.

Automation often overtook the rate of human progress, for instance in developing metatheorems. Does this create an opportunity cost? Raised as a possible discussion point here by Zoltan Kocsis.

14 Conclusions and Future Directions

TODO: Expand this sketch

Insights learned, future speculation. Utilize the discussion on future directions. Some ideas from that list:

- A database of triple implications (EquationX and EquationY imply EquationZ) - see also this discussion.
- Are there any equational laws that have no non-trivial finite models, but have surjunctive models?
- Can we produce interesting examples of irreducible implications EquationX -¿ EquationY (no intermediate EquationZ can interpose)
- Degree of satisfiability results, e.g., if a central groupoid obeys the natural central groupoid axiom 99% of the time, is it a natural central groupoid?
- Kisielewicz's question: does 5093 have any infinite models?
- "Insight mining" the large corpus of automated proofs that have been generated.
- How machine-learnable is the implication graph? (See AI/ML section)

Acknowledgments

Acknowledgments to the broader Lean Zulip community and smaller contributors not listed as authors.

A Numbering system

In this section we record the numbering conventions we use for equational laws.

For this formal definition we use the natural numbers $0, 1, 2, \dots$ to represent and order indeterminate variables; however, in the main text, we use the symbols $x, y, z, w, u, v, r, s, t$ instead (and do not consider any laws with more than eight variables).

We requiring extend the ordering on indeterminate variables to a well-ordering on words w in the free magma generated by these variables by declaring $w > w'$ if one of the following holds:

- w has a larger order than w'.
- $w = w_1 \diamond w_2$ and $w' = w'_1 \diamond w'_2$ have the same order $n \geq 1$ with $w_1 > w'_1$.
- $w = w_1 \diamond w_2$ and $w' = w'_1 \diamond w'_2$ have the same order $n \geq 1$ with $w_1 = w'_1$ and $w_2 > w'_2$.

Thus for instance

$$0 < 1 < 0 \diamond 0 < 0 \diamond 1 < 1 \diamond 0$$

and

$$1 \diamond 1 < 0 \diamond (0 \diamond 0) < (0 \diamond 0) \diamond 0.$$

We similarly place a well-ordering on equational laws $w_1 \simeq w_2$ by declaring $w_1 \simeq w_2 > w'_1 \simeq w'_2$ if one of the following holds: as follows:

- $w_1 \simeq w_2$ has a longer order than $w'_1 \simeq w'_2$.
- If $w_1 \simeq w_2$ has the same order as $w'_1 \simeq w'_2$, and $w_1 > w'_1$.
- If $w_1 \simeq w_2$ has the same order as $w'_1 \simeq w'_2$, $w_1 = w'_1$, and $w_2 > w'_2$.

Two equational laws are equivalent if one can be obtained from another by some combination of relabeling the variables and applying the symmetric law $w_1 \simeq w_2 \iff w_2 \simeq w_1$. For instance, $(0 \diamond 1) \diamond 2 \simeq 1$ is equivalent to $0 \simeq (1 \diamond 0) \diamond 2$. We then eliminate all equational laws that are not minimal in their equivalence class; for instance, we would eliminate $(0 \diamond 1) \diamond 2 \simeq 1$

in favor of $0 \simeq (1 \diamond 0) \diamond 2$. Finally, we eliminate any law of the form $w \simeq w$ other than $0 \simeq 0$. We then number the remaining equations $E1, E2, \ldots$. For instance, $E1$ is the trivial law $0 \simeq 0$, $E2$ is the constant law $0 \simeq 1$, $E3$ is the idempotent law $0 \simeq 0 \diamond 0$, and so forth. Lists and code for generating these equations, or the equation number attached to a given equation, can be found at the ETP repository.

The number of equations in this list of order $n = 0, 1, 2, \ldots$ is given by

$$2, 5, 39, 364, 4284, 57882, 888365, \ldots$$

(`https://oeis.org/A376640`). The number can be computed to be

$$C_{n+1} B_{n+2}/2$$

if n is odd, 2 if $n = 0$, and

$$(C_{n+1} B_{n+2} + C_{n/2}(2D_{n+2} - B_{n+2}))/2 - C_{n/2} B_{n/2+1}$$

if $n > 2$ is even, where C_n, B_n are the Catalan and Bell numbers, and D_n is the number of partitions of $[n]$ up to reflection, which for $n = 0, 1, 2, \ldots$ is

$$1, 1, 2, 4, 11, 32, 117, \ldots$$

(`https://oeis.org/A103293`). A proof of this claim can be found in the ETP blueprint. In particular, there are 4694 equations of order at most 4.

B Proofs of Theoretical Results

Provide the interesting proofs mentioned in the results section, while routine proofs can refer to the blueprint or Lean.

C Author Contributions

In this section we list the authors of this paper, grant support, and their contributions to this project, using the following standard CRediT categories:

1. Conceptualization: Ideas; formulation or evolution of overarching research goals and aims.
2. Data Curation: Management activities to annotate (produce metadata), scrub data and maintain research data (including software code, where it is necessary for interpreting the data itself) for initial use and later reuse.

3. Formal Analysis: Application of statistical, mathematical, computational, or other formal techniques to analyze or synthesize study data.

4. Funding Acquisition: Acquisition of the financial support for the project leading to this publication.

5. Investigation: Conducting a research and investigation process, specifically performing the experiments, or data/evidence collection.

6. Methodology: Development or design of methodology; creation of models.

7. Project Administration: Management and coordination responsibility for the research activity planning and execution.

8. Resources: Provision of study materials, reagents, materials, patients, laboratory samples, animals, instrumentation, computing resources, or other analysis tools.

9. Software: Programming, software development; designing computer programs; implementation of the computer code and supporting algorithms; testing of existing code components.

10. Supervision: Oversight and leadership responsibility for the research activity planning and execution, including mentorship external to the core team.

11. Validation: Verification, whether as a part of the activity or separate, of the overall replication/reproducibility of results/experiments and other research outputs.

12. Visualization: Preparation, creation and/or presentation of the published work, specifically visualization/data presentation.

13. Writing - Original Draft Preparation: Creation and/or presentation of the published work, specifically writing the initial draft (including substantive translation).

14. Writing – Review and Editing: Preparation, creation and/or presentation of the published work by those from the original research group, specifically critical review, commentary or revision – including pre- or post-publication stages.

- ...

- Terence Tao, Department of Mathematics, UCLA, tao@math.ucla.edu: Conceptualization, Data Annotation, Investigation, Methodology, Project Administration, Writing - Original Draft Preparation, Writing - Review and Editing. Supported by NSF grant DMS-2347850.

- ...

References

[1] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, Cambridge, 1998.

[2] Trevor Evans. Products of points—some simple algebras and their identities. *Amer. Math. Monthly*, 74:362–372, 1967.

[3] Timothy Gowers and Michael Nielsen. Massively collaborative mathematics. *Nature*, 461(7266):879–881, October 2009.

[4] Graham Higman and B. H. Neumann. Groups as groupoids with one law. *Publ. Math. Debrecen*, 2:215–221, 1952.

[5] A. Kisielewicz. Austin identities. *Algebra Universalis*, 38(3):324–328, 1997.

[6] Donald E. Knuth. Notes on central groupoids. *J. Combinatorial Theory*, 8:376–390, 1970.

[7] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, pages 263–297. Pergamon, Oxford-New York-Toronto, Ont., 1970.

[8] André Kündgen, Gregor Leander, and Carsten Thomassen. Switchings, extensions, and reductions in central digraphs. *J. Combin. Theory Ser. A*, 118(7):2025–2034, 2011.

[9] William McCune. Solution of the Robbins problem. *J. Automat. Reason.*, 19(3):263–276, 1997.

[10] William McCune. Single axioms: with and without computers. In *Computer mathematics (Chiang Mai, 2000)*, volume 8 of *Lecture Notes Ser. Comput.*, pages 83–89. World Sci. Publ., River Edge, NJ, 2000.

[11] William McCune, Robert Veroff, Branden Fitelson, Kenneth Harris, Andrew Feist, and Larry Wos. Short single axioms for Boolean algebra. *J. Automat. Reason.*, 29(1):1–16, 2002.

[12] Ralph McKenzie. On spectra, and the negative solution of the decision problem for identities having a finite nontrivial model. *J. Symbolic Logic*, 40:186–196, 1975.

[13] N. S. Mendelsohn and R. Padmanabhan. Minimal identities for Boolean groups. *J. Algebra*, 34:451–457, 1975.

[14] C. A. Meredith and A. N. Prior. Equational logic. *Notre Dame J. Formal Logic*, 9:212–226, 1968.

[15] J. D. Phillips and Petr Vojtˇ echovský. The varieties of loops of Bol-Moufang type. *Algebra Universalis*, 54(3):259–271, 2005.

[16] Alfred Tarski. Ein beitrag zur axiomatik der abelschen gruppen. *Fundamenta Mathematicae*, 30(1):253–256, 1938.

[17] Stephen Wolfram. The physicalization of metamathematics and its implications for the foundations of mathematics, Mar 2022.