Using proof assistants as more than checkers, and how to write papers about it

Yannick Forster

Inria, Cambium Team, Paris

Logic Mentoring Workshop at FloC 2026

Outline

- A survey of uses of proof assistants as true assistants, with a bias towards surprising authorship
- How to write papers about proof assistant use?

Four colour theorem (proof assistant as checker)

1852 Conjecture by Francis Guthrie

1879 Initial proof by Alfred Kempe

1890 Percy Heawood finds a mistake (5 colour theorem)

1976 Proof by Kenneth Appel and Wolfgang Haken

- Proof idea: find an "unavoidable", "reducible" set of configurations
- Reducibility: Checked by computer program, 15k lines on punch cards, took 2 days
- Unavoidability: 400 pages of microfiche, checked manually by Appel, Haken, and Haken's teenage daughter Dorothea Blostein

2004 Proof in Rocq by Georges Gonthier with Benjamin Werner

BB(5)

1955 Tibor Radó introduces "The busy beaver game":
BB(n) is the maximal number of steps a Turing machine with n states can take

Determining BB(24) determines the Collatz conjecture
BB(300) is independent from most foundations

1966 Allen Brady discovers 4 state machine taking 107 steps

1974 Brady proves BB(4) = 107
But there are 17 trillion possible 5 state Turing machines...

1989 Heiner Marxen and Jürgen Buntrock find 5 state machine taking 47,176,870 steps

2020 Scott Aaronson conjectures BB(5) = 47,176,870

2021 Busy beaver challenge started by Tristan Stérin

The Busy Beaver challenge

- Idea: terminating machines are easy, just run them
- Challenge comes from non-termination is the challenge, various proof techniques (think: cycle detection)
- All proofs need to be backed by independent executable non-termination checkers in different programming languages
- Overall: enumerate all possible machines efficiently, run them through checkers
- $\sim$800+ members on Discord, $\sim$50 contributors, mostly not professional mathematicians

CoqBB(5) (proof assistant as assistant)

- In 2023, only few "holdouts" left – but those are the hardest machines
- Maja Kadziolka in July 2023: "Hi there! I'm interested in formalizing the algorithms used to decide halting in a proof assistant such as Coq. Has anything like this been attempted before?", starts BusyCoq project
- In May 2024 anonymous contributor `mxdys` publishes full Coq proof, using BusyCoq as a library, solving all remaining holdouts
- I was asked to review the Coq proof by Tristan – took me about 10 minutes to verify that the statement is correct, did not have to check the proof
- Paper at STOC 2026 by The bbchallenge Collaboration, Justin Blanchard, Daniel Briggs, Konrad Deka, Nathan Fenner, Yannick Forster, Georgi Georgiev (Skelet), Matthew L. House, Rachel Hunter, Iijil, Maja Kadziolka, Pavel Kropitz, Shawn Ligocki, mxdys, Mateusz Naściszewski, savask, Tristan Stérin, Chris Xu, Jason Yuen, Théo Zimmermann

`https://www.quantamagazine.org/amateur-mathematicians-find-fifth-busy-beaver-turing-machine-20240702`

Hilbert's tenth problem

1900 Hilbert's tenth problem: *Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers.*

1944 Emil Post conjectures undecidability

1949 Martin Davis conjectures: A set is Diophantine if and only if it is r.e.

1960 Julia Robinson: Existence of exponential Diophantine set proves Davis' conjecture

1970 Yuri Matiyasevich proves that Fibonacci numbers are Diophantine

2018 R.e. sets are Diophantine in Isabelle/HOL: Deepak Aryal, Jonas Bayer, Bogdan Ciurezu, Marco David, Yiping Deng, Prabhat Devkota, Simon Dubischar, Malte Sophian Hassler, Yufei Liu, Maria Antonia Oprea, Abhik Pal, and Benedikt Stock

H10 in Isabelle (proof assistant as assistant)

2017 prize at the German high school research competition awarded to work on Hilbert's tenth problem, supervised by Prof. Dirk Schleicher, Bremen

2017 Yuri Matiyasevich visits Bremen and expresses interest in formal verification

2018 undergraduates form research group to formalise H10 in Isabelle/HOL

2019 unrelated project finishes proof in Rocq

2019 Isabelle proof that r.e. sets are Diophantine in Isabelle/HOL: Deepak Aryal, Jonas Bayer, Bogdan Ciurezu, Marco David, Yiping Deng, Prabhat Devkota, Simon Dubischar, Malte Sophian Hassler, Yufei Liu, Maria Antonia Oprea, Abhik Pal, and Benedikt Stock

2020 undergraduate research group formed in Paris to develop and formalise novel results on universal bounds – "in-situ / parallel formalisation"

2025 improved bounds paper on arxiv: Jonas Bayer, Marco David, Malte Hassler, Yuri Matiyasevich, Dierk Schleicher, Thomas Serafini, Simon Dubischa, Timothé Ringeard, Xavier Pigé, Anna Danilkin, Annie Yao, Mathis Bouverot-Dupuis, Paul Wang, Quentin Vermande, Theo André, Loïc Chevalier, Charlotte Dorneich, Eva Brenner, Zhengkun Ye, Kevin Lee

Proof assistants increase accessibility
even and especially for people outside classic academic paths

Venues for publication of results

- Novel mathematical results with formalisation:
  - Annals of formalised Math journal (AfM)
- Proof assistant use, including novel or known formalised results:
  - Interactive Theorem Proving (ITP)
  - Certified Programs and Proofs (CPP)
- Results related to logic or logical foundations:
  - Computer Science Logic (CSL)
  - Formal Structures for Computation and Deduction (FSCD)
  - Logic in Computer Science (LICS)
  - Logical Methods in Computer Science journal (LMCS)
- Results related to implementation
  - International Joint Conference on Automated Reasoning (IJCAR)
  - Computer Aided Verification (CAV)
  - Theory and Applications of Satisfiability Testing (SAT)

Venues for publication of results

- Results related to programming languages or type theoretic foundations:
  - Principles of Programming Languages (POPL)
  - International Conference on Functional Programming (ICFP)
  - Programming Languages Design and Implementation (PLDI)
  - European Symposium on Programming (ESOP)

Advice for papers using proof assistants

Sources

Andrej Bauer, *How to review formalized mathematics*, 2013.
`https://math.andrej.com/2013/08/19/how-to-review-formalized-mathematics/`
Assia Mahboubi, *Checking machine-checked proofs*, 2017.
`https://project.inria.fr/coqexchange/checking-machine-checked-proofs/`

Virtues of a formalization project

Not every project will excel in all of these criteria, and they are not requirements for a paper to be published in the AFM. Nevertheless we encourage authors and reviewers to take these points into consideration.

- Novelty of the formalization: Has this topic been formalized before? If so, in which proof assistant ecosystems? How does the current formalization relate to existing ones?
- Novelty of the mathematics: Is the subject of the formalization a new or recent result? If so, how significant is the new result? If not, what is the impact and importance of the known result? Is the importance properly discussed in the paper?
- Insightfulness: Did the formalization yield any new mathematical insights? Did it yield any general lessons about how mathematical concepts can or should be represented formally?
- Generality: Does the project take on a specific mathematical topic in detail, or does it work at a high level of abstraction? How widely do the results apply?
- Integration: Does the project build on existing libraries? Has it or will it be integrated into these libraries? How interconnected is it with other parts of the library? Is it incompatible with existing libraries or parts of them?
- Reproducibility: Does this work teach lessons that can be applied in other libraries or proof assistant ecosystems?
- How complex, long, or difficult is the formalization itself?
- Did the choice of proof assistant or logical foundation crucially influence the formalization process?
- How readable and well-documented is the code artifact?

We emphasize a focus on the usability of formalization projects: authors are encouraged to explain the interfaces around their definitions, if applicable with examples of how their development can be used, and to provide all their files with adequate documentation.

Separating the maths from the formalisation

- Use mathematical notation in mathematical sections
- Separate sections / subsection on how concepts are captured in proof assistant
- Code-heavy papers will almost always be rejected: one reviewer will not be familiar with this proof assistant
- Make theorems in pdf hyperlinked to html of code and annotate code with lemmas / theorems from paper
- No need to advocate use of PAs at mentioned venues!
- Inaccessible paper for member of other PA community is *reason for rejection*

Advice by Steven Weinberg and Stephen Hawking

1. You don't have to know everything.
2. Aim for rough water.
3. Forgive yourself for wasting time.
4. Know the history of (your) science.

https://www.nature.com/articles/426389a

"Bruce, here's some advice: The problem with physics is that most of the days we don't make any major headway (on our projects). That's why you should do other stuff: listen to music, meet good friends. There's one exception to this rule: If you find a solution for a given problem, you work 24 hours a day and forget everything else. Until the problem is solved in its entirety."

Relation to other formalisation projects?

- Reviewers expect a *thorough* comparison with related formalisation work
- Related work needs to cover *all other* proof assistants!
- Nobody is an expert in all proof assistants, so understanding contributions of related work is labour-intensive and takes time
- More than just listing: Compare approach, proof structure, and design decisions. Comparison does not have to be favorable in all aspects!
- Check libraries, github, workshops, conference proceedings.
- The PA community is small: reviewers often the authors of related work, and choosing the most cited related work as external reviewer is common
- Insufficient discussion of related work is *reason for rejection*

On citing

- Cite by name to avoid bias ("integral" or "narrative" citations)
- Avoid using first names when citing
- Cite first work that introduced a concept, not to the most recent or textbooks.
- Use doi.org bib entries, or dblp if necessary – in any case add a doi!
- Sentences still need to make sense after [citation] is removed – read it out loud!

Insight created from formalisation?

- What did the authors learn that they didn't know before?
- Typical insights:
  - A textbook proof was wrong, incomplete, or used unstated assumptions
  - A simpler or more general proof emerged from the formalisation
  - An algorithm was extracted from a proof
  - An implicit use of the axiom of choice or similar was made explicit
- The "insight" can be on an engineering level, or didactic
- Lack of insight from formalisation lowers significance of contribution
- Lacking discussion of insight is *reason for rejection*

Some style advice

- Use "we" sparingly.
- Can mean "the paper" more than any actual human
- Can mean "we, the authors" when discussing insight or beliefs
- Can mean "we, authors and readers together" in technical passages
- "We define" when introducing somebody else's concept is confusing.

Writing advice by Steve Zdancewic
`https://www.cis.upenn.edu/~stevez/writing-tips.html`

The artifact matters

- Submit the proof artifact as zip at submission time
- Make sure there is a README with *exact, detailed* instructions how to set up the environment
- Version number of proof assistant + all used libraries
- Include build commands (`make`, `lake build`, etc.)
- List all axioms, standard and non-standard ones, and suggest commands in README that the reviewers can use to review axioms `Print Assumptions` / `#print axioms`
- No `Admitted`, `sorry`, etc, not even in comments
- Turn gaps into assumptions, and make them transparent and prominent
- Make final theorem statement reviewable without proof, e.g.
  - `Preliminaries.v`
  - `TheoremStatement.v` only importing `Preliminaries.v`
  - `Proof.v`, etc
  - `TheoremProof.v`
- Is the formalisation integrated into an existing library, or standalone?

Stanislaw Ulam in 1964

"Although to many people the electronic computer has come to symbolize the importance of mathematics in the modern world, few professional mathematicians are closely acquainted with the machine. Some, in fact, seem even to fear that individual scientific efforts will be pushed into the background or replaced by less imaginative, purely mechanical habits of research. I believe such fears to be quite groundless. It is preferable to regard the computer as a handy device for manipulating and displaying symbols. Even the most abstract thinkers agree that the simple act of writing down a few symbols on a piece of paper facilitates concentration. In this respect alone – and it is not a trivial one – the new electronic machines enlarge our effective memory and provide a marvelous extension of the means for experimenting with symbols in science."

https://www.jstor.org/stable/24931634

How complex was this?

- Quantitative data:
  - Lines of code (spec vs. proofs vs. automation)
  - Compilation time
  - Degree of automation
- Compare to the length of the textbook proof: the de Bruijn factor
- Community doesn't have a standard yet. E.g. some argue that we have to compare gzipped data.
- How long was the process? Experience level of authors? Learning curve?
- Often short, elegant formalisations are a harder sell than long, complicated ones.

"Simplicity does not precede complexity, but follows it."

Alan Perlis

"Making the simple complicated is commonplace.
Making the complicated simple – awesomely simple – that's creativity."

Charles Mingus

Did the choice of proof assistant matter?

- This is *not* about marketing your favourite tool
- Logical foundations matter:
  - Universe polymorphism, sort polymorphism, proof by computation
  - Constructive vs. classical, predicative vs. impredicative
  - Univalence, HoTT, set-theoretic foundations
  - Would the proof work in a different system? Why or why not?
- Practical tooling matters too:
  - Automation (tactics, decision procedures), library ecosystem
  - IDE support, compilation times, community support
- Honest assessments are more valuable than advocacy

Georg Cantor

"In mathematics the art of proposing a question must be held of higher value than solving it."

Sources

Andrej Bauer, *How to review formalized mathematics*, 2013.
`https://math.andrej.com/2013/08/19/how-to-review-formalized-mathematics/`
Assia Mahboubi, *Checking machine-checked proofs*, 2017.
`https://project.inria.fr/coqexchange/checking-machine-checked-proofs/`

Virtues of a formalization project

Not every project will excel in all of these criteria, and they are not requirements for a paper to be published in the AFM. Nevertheless we encourage authors and reviewers to take these points into consideration.

- Novelty of the formalization: Has this topic been formalized before? If so, in which proof assistant ecosystems? How does the current formalization relate to existing ones?
- Novelty of the mathematics: Is the subject of the formalization a new or recent result? If so, how significant is the new result? If not, what is the impact and importance of the known result? Is the importance properly discussed in the paper?
- Insightfulness: Did the formalization yield any new mathematical insights? Did it yield any general lessons about how mathematical concepts can or should be represented formally?
- Generality: Does the project take on a specific mathematical topic in detail, or does it work at a high level of abstraction? How widely do the results apply?
- Integration: Does the project build on existing libraries? Has it or will it be integrated into these libraries? How interconnected is it with other parts of the library? Is it incompatible with existing libraries or parts of them?
- Reproducibility: Does this work teach lessons that can be applied in other libraries or proof assistant ecosystems?
- How complex, long, or difficult is the formalization itself?
- Did the choice of proof assistant or logical foundation crucially influence the formalization process?
- How readable and well-documented is the code artifact?

We emphasize a focus on the usability of formalization projects: authors are encouraged to explain the interfaces around their definitions, if applicable with examples of how their development can be used, and to provide all their files with adequate documentation.