# A Short Survey of Weak Reasoning Assistants

MATTHEW SOTOUDEH, Stanford University, USA

Research on reasoning assistants tends to focus on tools that can mechanically verify fully formalized, rigorous proofs [3, 6]. In practice, however, little mathematics is formalized to such a degree. Instead, DeMillo et al. [7] argue that proof is a *social process* meant to convince, communicate, and refine a shared understanding or insight with other human mathematicians [2].

This extended abstract uses the term *weak reasoning assistants* to unify and survey a scattered collection of work on tools and techniques that assist mathematicians with that less-than-formal task of proof without requiring a fully mechanized argument. While even the pencil might be seen as a *very* weak reasoning assistant, we focus on the stronger and less well-known middle of the spectrum: techniques that know *something* about the underlying mathematics without expecting to verify entire deductive arguments. Such assistants often result from either adding structure to an otherwise unstructured tool, or removing structure from a more structured one.

For example, structured proofs [14–16] add more structure to LaTeX documents [13] by requiring deductive steps and justifications be made explicit. Formal proof sketches [29] are dual to this, where steps are removed from a fully-mechanized proof to create a sketch that is better suited to human communication. In between, Weak Type Theory [21] enforces the linguistic form of viable arguments while ignoring logical correctness of their contents.

The Penrose language [31] incorporates semantic knowledge about set theory to greatly simplify visualization of mathematical claims compared to more unstructured, point-oriented tools like TikZ [27]. Similar domain-specific visualization tools have shown promise in building interactive arguments using web technologies [23].

Computer algebra systems like Mathematica [30] and Maple [19] understand semantics of individual operations, such as polynomial multiplication. The overall deductive argument, however, is often only expressed in prose referring to the code [26]. Mathematicians generally find such tools useful to perform tedious symbolic manipulations, search for counterexamples to conjectures, and explore finite instances of structures to gain or communicate intuition [5].

Property-based testing for Isabelle [4] and Coq [8] automatically searches for counterexamples to conjectured theorem statements. Aligned with Youssef [32], Greiner-Petter et al. [9] propose a tool to automatically search for counterexamples to LaTeX equations parsed from online repositories. Failure to find a counterexample can be useful heuristic evidence for the claim. Computer searches have been used to find both counterexamples to [17] and full proofs of [1, 10] conjectures. Related tools help refine models of programming languages [12] and system designs [11].

We expect to see weak reasoning assistants continue to bring mechanical assistance to everyday mathematicians, analogous to lightweight formal methods in traditional software engineering [24]. One mathematician has already proposed a tool for linting LaTeX documents [18]. Dually, we suggest a community effort towards a theorem prover that explicitly prioritizes ease-of-use over small kernels and soundness-at-all-costs, e.g., by encouraging the use of complicated decision procedures to establish certain facts, even if the procedure does not produce a proof checkable by the underlying type checker. To a limited extent this approach is taken by Mizar [20, §3], [28, §7], but we are not aware of a free software proof system with similar focus. A gradually-mechanizing assistant might unify unchecked structured proofs with machine-checkable formal proofs, allowing portions of an argument to be checked modulo others analogous to work in gradual typing [25] and typed holes [22] for programming languages.

Author's address: Matthew Sotoudeh, Stanford University, Stanford, CA, USA, sotoudeh@stanford.edu.

## ACKNOWLEDGMENTS

The author is grateful for discussions with David K. Zhang, Evan Laufer, and Dawson Engler; financial support under National Science Foundation grants 1918056 and DGE-1656518; and feedback of the anonymous reviewers that greatly improved the quality of this extended abstract.

## REFERENCES

[1] Kenneth I Appel and Wolfgang Haken. 1989. *Every planar map is four colorable.* American Mathematical Society. 0821851039

[2] Jonas Bayer, Christoph Benzmüller, Kevin Buzzard, Marco David, Leslie Lamport, Yuri Matiyasevich, Lawrence Paulson, Dierk Schleicher, Benedikt Stock, and Efim Zelmanov. 2022. Mathematical Proof Between Generations. (2022). https://doi.org/10.48550/ARXIV.2207.04779

[3] Robert S. Boyer. 1994. A mechanically proof-checked encyclopedia of mathematics: Should we build one? Can we?. In *12th International Conference on Automated Deduction*, Alan Bundy (Ed.). https://doi.org/10.1007/3-540-58156-1_17

[4] Lukas Bulwahn. 2012. The New Quickcheck for Isabelle - Random, Exhaustive and Symbolic Testing under One Roof. In *Certified Programs and Proofs - Second International Conference, CPP 2012. Proceedings*, Chris Hawblitzel and Dale Miller (Eds.). https://doi.org/10.1007/978-3-642-35308-6_10

[5] Andrea Bunt, Michael A. Terry, and Edward Lank. 2009. Friend or Foe?: Examining CAS Use in Mathematics Research. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009*, Dan R. Olsen Jr., Richard B. Arthur, Ken Hinckley, Meredith Ringel Morris, Scott E. Hudson, and Saul Greenberg (Eds.). https://doi.org/10.1145/1518701.1518740

[6] Thierry Coquand and Gérard P. Huet. 1988. The Calculus of Constructions. *Information and Computation* 76, 2/3 (1988), 95–120. https://doi.org/10.1016/0890-5401(88)90005-3

[7] Richard A. DeMillo, Richard J. Lipton, and Alan J. Perlis. 1977. Social Processes and Proofs of Theorems and Programs. In *Conference Record of the Fourth ACM Symposium on Principles of Programming, January 1977*, Robert M. Graham, Michael A. Harrison, and Ravi Sethi (Eds.). https://doi.org/10.1145/512950.512970

[8] Maxime Dénès, Catalin Hritcu, Leonidas Lampropoulos, Zoe Paraskevopoulou, and Benjamin C Pierce. 2014. QuickChick: Property-Based Testing for Coq. In *The Coq Workshop.* https://catalin-hritcu.github.io/talks/coq6_submission_4.pdf Accessed: 2022-10-13.

[9] André Greiner-Petter, Howard S. Cohl, Abdou Youssef, Moritz Schubotz, Avi Trost, Rajen Dey, Akiko Aizawa, and Bela Gipp. 2022. Comparative Verification of the Digital Library of Mathematical Functions and Computer Algebra Systems. In *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Proceedings, Part I*, Dana Fisman and Grigore Rosu (Eds.). https://doi.org/10.1007/978-3-030-99524-9_5

[10] Marijn J. H. Heule and Oliver Kullmann. 2017. The Science of Brute Force. *Communications of the ACM* 60, 8 (2017), 70–79. https://doi.org/10.1145/3107239

[11] Daniel Jackson. 2019. Alloy: a Language and Tool for Exploring Software Designs. *Communications of the ACM* 62, 9 (2019), 66–76. https://doi.org/10.1145/3338843

[12] Casey Klein, John Clements, Christos Dimoulas, Carl Eastlund, Matthias Felleisen, Matthew Flatt, Jay A. McCarthy, Jon Rafkind, Sam Tobin-Hochstadt, and Robert Bruce Findler. 2012. Run your research: on the effectiveness of lightweight mechanization. In *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012*, John Field and Michael Hicks (Eds.). https://doi.org/10.1145/2103656.2103691

[13] Leslie Lamport. 1994. *LaTeX - A Document Preparation System: User's Guide and Reference Manual, Second Edition.* Pearson / Prentice Hall. 978-0-201-52983-8

[14] Leslie Lamport. 1995. How to Write a Proof. *The American Mathematical Monthly* 102, 7 (1995), 600–608. https://doi.org/10.2307/2974556

[15] Leslie Lamport. 2011. *The* `pf2` *Package.* https://lamport.azurewebsites.net/latex/pf2.pdf Accessed: 2022-10-12.

[16] Leslie Lamport. 2012. How to write a 21st century proof. *Journal of Fixed Point Theory and Applications* 11, 1 (2012), 43–63. https://doi.org/10.1007/s11784-012-0071-6

[17] L.J. Lander and T.R. Parkin. 1966. Counterexample to Euler's conjecture on sums of like powers. *Bulletin of the Amererican Mathematical Society* 72, 6 (1966), 1079. https://doi.org/10.1090/S0002-9904-1966-11654-3

[18] Richard Lipton. 2015. Lint For Math. https://rjlipton.wpcomstaging.com/2015/03/08/lint-for-math/ Accessed: 2022-10-13.

[19] Michael B. Monagan, Keith O. Geddes, K. Michael Heal, George Labahn, Stefan M. Vorkoetter, James McCarron, and Paul DeMarco. 1998. *Maple V Programming Guide.* Maplesoft, Waterloo ON, Canada. 978-1-4612-2214-9 https://doi.org/10.1007/978-1-4612-2214-9

[20] Adam Naumowicz and Czeslaw Bylinski. 2004. Improving Mizar Texts with Properties and Requirements. In *Mathematical Knowledge Management, Third International Conference, MKM 2004, Proceedings*, Andrea Asperti, Grzegorz

Bancerek, and Andrzej Trybulec (Eds.). https://doi.org/10.1007/978-3-540-27818-4_21

[21] Robert Pieter Nederpelt. 2002. *Weak Type Theory: a formal language for mathematics.* Technical Report. Technische Universiteit Eindhoven. https://pure.tue.nl/ws/portalfiles/portal/2423166/200205.pdf Accessed: 2022-10-13.

[22] Cyrus Omar, Ian Voysey, Ravi Chugh, and Matthew A. Hammer. 2019. Live Functional Programming with Typed Holes. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 14:1–14:32. https://doi.org/10.1145/3290327

[23] Steven Petryk. 2022. Mafs: React components for interactive math. https://mafs.dev/ Accessed: 2022-10-13.

[24] Alastair Reid, Luke Church, Shaked Flur, Sarah de Haas, Maritza Johnson, and Ben Laurie. 2020. Towards making formal methods normal: meeting developers where they are. In *Human Aspects of Types and Reasoning Assistants.* https://doi.org/10.48550/ARXIV.2010.16345

[25] Jeremy G. Siek, Michael M. Vitousek, Matteo Cimini, and John Tang Boyland. 2015. Refined Criteria for Gradual Typing. In *1st Summit on Advances in Programming Languages, SNAPL 2015*, Thomas Ball, Rastislav Bodík, Shriram Krishnamurthi, Benjamin S. Lerner, and Greg Morrisett (Eds.). https://doi.org/10.4230/LIPIcs.SNAPL.2015.274

[26] Jeremy Singer. 2020. Notes on notebooks: is Jupyter the bringer of jollity?. In *Proceedings of the 2020 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, Onward! 2020.* https://doi.org/10.1145/3426428.3426924

[27] Till Tantau. 2022. *TikZ & PGF.* https://pgf-tikz.github.io/pgf/pgfmanual.pdf Accessed: 2022-10-13.

[28] Freek Wiedijk. 1999. Mizar: An Impression. (1999). https://cs.ru.nl/~freek/mizar/mizarintro.pdf Accessed: 2022-10-13.

[29] Freek Wiedijk. 2003. Formal Proof Sketches. In *Types for Proofs and Programs, International Workshop, TYPES 2003, Revised Selected Papers*, Stefano Berardi, Mario Coppo, and Ferruccio Damiani (Eds.). https://doi.org/10.1007/978-3-540-24849-1_24

[30] Stephen Wolfram. 2000. *The Mathematica Book.* Wolfram Research Inc. 0965053202

[31] Katherine Ye, Wode Ni, Max Krieger, Dor Ma'ayan, Jenna Wise, Jonathan Aldrich, Joshua Sunshine, and Keenan Crane. 2020. Penrose: From Mathematical Notation to Beautiful Diagrams. *ACM Transactions on Graphics* 39, 4 (2020), 144. https://doi.org/10.1145/3386569.3392375

[32] Abdou Youssef. 2017. Part-of-Math Tagging and Applications. In *Intelligent Computer Mathematics - 10th International Conference, CICM 2017, Proceedings*, Herman Geuvers, Matthew England, Osman Hasan, Florian Rabe, and Olaf Teschke (Eds.). https://doi.org/10.1007/978-3-319-62075-6_25