Learning plausible and useful conjectures

Albert Q. Jiang, Wenda Li, and Mateja Jamnik

University of Cambridge
{qj213, wl302, mj201}@cam.ac.uk

Abstract

Conjecturing is an important activity in mathematics. In this paper, we look at the why and the how of using machine learning to generate mathematical conjectures. We argue that (1) conjecturing is beneficial, both practically and theoretically; (2) conjecture learning should make use of available premises and goals in theorems. We also deliver some design considerations for building an automated conjecturer.

1 Conjecturing as an essential mathematical activity

Consider lemma construction in theorem proving: in the course of proving a theorem, one might realise that a particular conjecture, if true, makes it easier to prove the original theorem. Once the conjecture is proved, it becomes a lemma. A well-known example of this is the proof of the Taniyama–Shimura–Weil conjecture, which implied Fermat's last theorem [Wil95]. Mathematical discovery is interleaved with finding interesting conjectures, and proving, refuting, or revising them [Lak15].

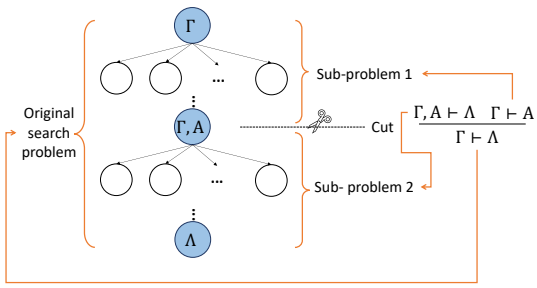

Figure 1: Theorem proving as search. The cut transforms the original search problem into two sub-problems by creating intermediate goal A.

From a computational perspective, formal theorem proving can be viewed as a search problem of finding the goal Λ given the premise Γ, as illustrated in Figure 1. A conjecture is then analogous to an intermediate goal (a *"cut"*). Cuts reduce the search space size exponentially w.r.t. their depth and therefore simplify the search [Boo84, CS97].

Conjecture/lemma construction has received relatively little attention from the AI community, considering its essential role in mathematics (see Appendix for a review). This is partially due to the difficulty of conjuring them: the space of possible conjectures is infinite. If we limit their size, the space is still combinatorially large and a universal heuristic for good conjectures is hard to find. Language models [BHA+21] sample from combinatorial spaces and have shown promising reasoning capabilities like solving maths competition problems [PHZ+22]. They can be easily configured to learn a general heuristic from human data. They are also not restricted to generating conjectures syntactically close to the premises or the goals. Thus, they complement symbolic reasoners, and are an obvious candidate for the task of conjecturing.

2 A quantitative metric for conjecturers

Before embarking on the specifics of a conjecturer, a metric for measuring the quality of conjectures is needed. We can examine conjectures qualitatively but that can be costly, hence not suited to large-scale experiments. For a quantitative metric, we should consider that ineffective conjectures can be true but not easily provable, untrue but not easily refutable, or trivially true

and useless for our goal. Meanwhile, effective conjectures usually unlock multiple theorems. In summary, the metric should prefer conjectures that are more provable, useful, and general. [CBW00] also used these criteria to measure how interesting mathematical discoveries are.

Suppose we have a conjecturer $\mathcal{C}$ and a base prover $\mathcal{B}$ (which could take the form of Sledgehammer [PB10] or a learning-assisted prover like LISA [JLHW21]). We can use $\mathcal{C}$ to propose new subgoals, and $\mathcal{B}$ to close them. The performance of conjecturers is measured by *the proportion of theorems proven this way, subtracting the proportion of theorems proven with $\mathcal{B}$ alone*. We can calculate a vector of this value indexed by the choice of $\mathcal{B}$. The vector is then a quantitative measure of the conjecturer's performance.

In this paper we argue that given the proposed metric, **conjecture learning should make use of available premises and goals of theorems.** Premises constrain the variable space in ways that are of interest for mathematics (e.g., the premise *p is prime* limits us to a small but interesting set of natural numbers). Focusing on these special variable spaces, one is more likely to make conjectures of relevance to human mathematics. This can increase the generality of conjectures: conjectures syntactically or semantically related to goals have a better chance of helping to prove them, instead of being trivially true (e.g., $0 = 0$). Conditioning on goals can improve the utility of conjectures. In the next section we detail some design considerations.

3 Building an automated conjecturer

The data and the environment Plenty of mathematical corpora and interactive environments are available: lean-gym [PHZ+22] for Lean, PISA [JLHW21] for Isabelle, coq-gym [YD19] for Coq, mlCoP [KUMO18] for Mizar, etc. Inside these proof corpora there are examples of conjecturing (Isabelle and Mizar have more due to their declarative proof style). Behaviour cloning can be deployed on these human conjecturing examples to bootstrap the conjecturer. For each datapoint, we should have the input-output pair to be in the following format: (input) the premises of the current problem; the goals of the current problem; the proof so far; (output) the conjecture written by the human.

The training loop Behaviour cloning alone does not guarantee a hugely useful conjecturer [LYWP21]. Since the conjecturer is bootstrapped from human conjecture examples, it is not aware of the ability of the base prover $\mathcal{B}$ and might propose conjectures that are too hard for it (we presume that the base prover $\mathcal{B}$ is weaker than a human). To deal with this problem, we should adapt our system to come up with conjectures that are both useful and *can be proven by $\mathcal{B}$*. For a set of problems, run the following procedure until convergence: use $\mathcal{C}$ and $\mathcal{B}$ to prove them; for failed proofs, filter out refutable conjectures with counter-example finding tools like `quickcheck` and `nitpick` (filtering out refutable conjectures with Isabelle tools was experimented by [NP18]); re-run $\mathcal{C}$ to refine the goals until they are either proved by $\mathcal{B}$ or a recursion depth limit is reached; collect all successful proofs as the gold-standard data; fine-tune both the conjecturer and the base prover on the gold-standard data. This procedure ensures that the conjecturer is not over-ambitious by only including provable conjectures in the gold-standard data. As the base prover improves via expert iteration [PHZ+22], we should expect the theorems proven and the conjectures created to become more and more advanced.

The network architecture We want to leverage the recent advances in learning-assisted theorem proving. As language model (LM)-based systems have demonstrated their potentials on multiple theorem provers [PS20, PHZ+22, JLHW21], textual information as input is preferred for generality. It is clear that the conjecturer $\mathcal{C}$ and the base prover $\mathcal{B}$ may share many common capabilities, therefore a network architecture that reflects this can be advantageous.

References

[BHA+21] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.

[Boo84] George Boolos. Don't eliminate cut. *Journal of Philosophical Logic*, pages 373–378, 1984.

[BSŻC93] Rajiv Bagai, Vasant Shanbhogue, JM Żytkow, and Shang-Ching Chou. Automatic theorem generation in plane geometry. In *International Symposium on Methodologies for Intelligent Systems*, pages 415–424. Springer, 1993.

[Bun88] Alan Bundy. The use of explicit plans to guide inductive proofs. In *International conference on automated deduction*, pages 111–120. Springer, 1988.

[CBW99] Simon Colton, Alan Bundy, and Toby Walsh. Automatic concept formation in pure mathematics. In *The 16th international joint conference on Artificial Intelligence - IJCAI '99*, 1999.

[CBW00] Simon Colton, Alan Bundy, and Toby Walsh. On the notion of interestingness in automated mathematical discovery. *International Journal of Human-Computer Studies*, 53(3):351–375, 2000.

[CS97] Alessandra Carbone and S Semmes. Making proofs without modus ponens: An introduction to the combinatorics and complexity of cut elimination. *Bulletin of the American Mathematical Society*, 34(2):131–159, 1997.

[DL82] Randall Davis and Douglas B Lenat. *Knowledge-Based Systems in Artificial Intelligence: 2 Case Studies*. McGraw-Hill, Inc., 1982.

[DVB+21] Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600(7887):70–74, 2021.

[Eps87] Susan L Epstein. On the discovery of mathematical theorems. In *IJCAI*, pages 194–197, 1987.

[Eps88] Susan L Epstein. Learning and discovery: One system's search for mathematical knowledge. *Computational Intelligence*, 4(1):42–53, 1988.

[Faj88] Siemion Fajtlowicz. On conjectures of graffiti. *Discret. Math.*, 72(1-3):113–118, 1988.

[IB96] Andrew Ireland and Alan Bundy. Productive use of failure in inductive proof. In *Automated Mathematical Induction*, pages 79–111. Springer, 1996.

[JLHW21] Albert Qiaochu Jiang, Wenda Li, Jesse Michael Han, and Yuhuai Wu. Lisa: Language models of isabelle proofs. In *Proceedings of the 6th Conference on Artificial Intelligence and Theorem Proving, AITP 2021, 5-11 September 2021, Aussois, France*, 2021.

[KUMO18] Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Miroslav Olsák. Reinforcement learning of theorem proving. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8836–8847, 2018.

[Lak15] Imre Lakatos. *Proofs and refutations: The logic of mathematical discovery*. Cambridge university press, 2015.

[Len76] Douglas Bruce Lenat. *AM: an artificial intelligence approach to discovery in mathematics as heuristic search.* Stanford University, 1976.

[LYWP21] Wenda Li, Lei Yu, Yuhuai Wu, and Lawrence C. Paulson. Isarstep: a benchmark for high-level mathematical reasoning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[NP18] Yutaka Nagashima and Julian Parsert. Goal-oriented conjecturing for isabelle/hol. *CoRR*, abs/1806.04774, 2018.

[NPW02] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.

[PB10] Lawrence C. Paulson and Jasmin Christian Blanchette. Three years of experience with sledgehammer, a practical link between automatic and interactive theorem provers. In Geoff Sutcliffe, Stephan Schulz, and Eugenia Ternovska, editors, *The 8th International Workshop on the Implementation of Logics, IWIL 2010, Yogyakarta, Indonesia, October 9, 2011*, volume 2 of *EPiC Series in Computing*, pages 1–11. EasyChair, 2010.

[PHZ+22] Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. *arXiv preprint arXiv:2202.01344*, 2022.

[PS20] Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

[RGM+21] Gal Raayoni, Shahar Gottlieb, Yahel Manor, George Pisha, Yoav Harris, Uri Mendlovic, Doron Haviv, Yaron Hadad, and Ido Kaminer. Generating conjectures on fundamental constants with the ramanujan machine. *Nature*, 590(7844):67–73, 2021.

[Rud92] Piotr Rudnicki. An overview of the mizar project. In *Proceedings of the 1992 Workshop on Types for Proofs and Programs*, pages 311–330, 1992.

[RWC+19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[UJ20] Josef Urban and Jan Jakubův. First neural conjecturing datasets and experiments. In *International Conference on Intelligent Computer Mathematics*, pages 315–323. Springer, 2020.

[Wan60] Hao Wang. Toward mechanical mathematics. *IBM J. Res. Dev.*, 4(1):2–22, 1960.

[Wil95] Andrew Wiles. Modular elliptic curves and fermat's last theorem. *Annals of mathematics*, 141(3):443–551, 1995.

[YD19] Kaiyu Yang and Jia Deng. Learning to prove theorems via interacting with proof assistants. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6984–6994. PMLR, 2019.

Appendix: a review of mathematical conjecturers

In a 1960 piece, Wang pointed out that making interesting conjectures is less easily mechanisable than formalising proofs [Wan60]. Indeed, compared with the research on automated theorem proving, conjecturing has received much less attention. Here we look at some prior works on mechanising mathematical conjecturing, including both symbolic and learning-based methods.

[Len76, DL82] described the **AM program**, which can reinvent important concepts in set theory and number theory, given basic facts such as sets and bags. AM is able to conjecture generalisations of existing concepts, among the discovery of other concepts, based on 242 heuristics. [Eps87, Eps88] detailed the **GT program** , which does concept forming, conjecture making, and theorem proving in graph theory. Graphs are very carefully represented such that efficient automation of these activities can be done. **The Graffiti program** [Faj88] maked numerical conjecutres on graph theory. Whenever a conjecture is made, the program tries to refute them using a database of graphs. Those that were not refuted were left as the final conjectures. **Bagai et. al's system** [BSŻC93] made and proved conjectures in plane geometry of the form that certain diagrams cannot be constructed. **The HR program** [CBW99] applied to many finite algebras, as well as number theory and graph theory. HR used seven production rules to find new concepts from old ones.

One common feature of these programs is that their domains of applications are relatively narrow. This is due to that representations of mathematical concepts are very different across different domains. It is thus difficult to design a symbolic algorithm to find new concepts that have a wide range of application areas. **The Ramanujan machine** [RGM+21] conjectures polynomial continued fractions that equate to fundamental constants, and **Davies et. al's system** [DVB+21] hints mathematicians about important relations in knot theory and representation theory. Although using learning, their representations are also very hard to extend.

Works that relate the most to our paper are **PGT** [NP18], **proof planners** [Bun88] and **critics** [IB96]. PGT [NP18] generates conjectures by mutating the goals and uses multiple filters to make sure that they were useful and not easily refutable. This approach requires the conjecture to lead directly to the goal (the gap can be closed by fastforce). Proof planners [Bun88] specify the high-level structure of a proof and proof critics [IB96] try to come up with useful conjectures from failed proofs. Both utilise the proof premises and goals extensively. These three methods all have a formal logic backend and thus are potentially very general. But they all require the conjectures to be similar to the premises or goals of the theorem, while the conjecture that is half way between them is the most effective at reducing the size of the search space. Proof planners and critics also need explicit instructions in the meta-logic of the proof assistant and can be costly to deploy.

A refreshing attempt was **Urban and Jakubův's system** [UJ20], where they fed theorems in Mizar [Rud92] in textual form to a GPT-2 style transformer [RWC+19] and directly sampled new theorem statements from it. However, the sampling was purely unconditional, so the generated statements could be seen as random extrapolations of other theorems in the latent space. Unsurprisingly, most generated theorems ended up quite trivial; how they related to other theorems, if at all, was entirely opaque. The **IsarStep** dataset [LYWP21] consists of intermediate conjectures in the Isabelle proof assistant [NPW02], but it suffers from requiring conjectures to be equivalent to the ground truth, when multiple equally valid proofs may have non-equivalent conjectures.