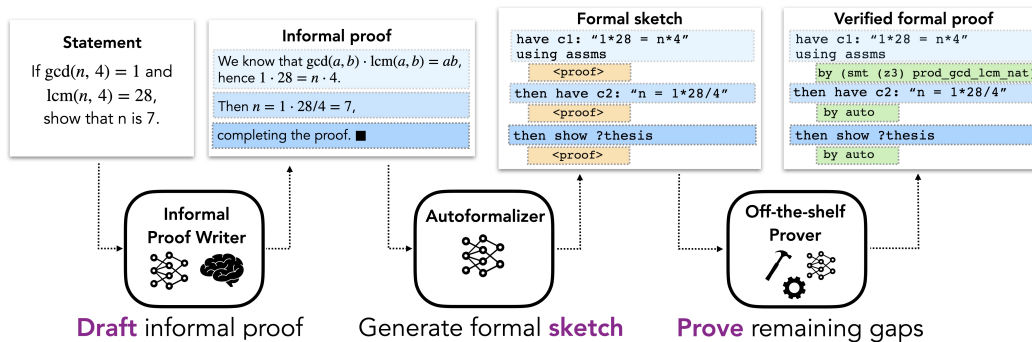

Draft, Sketch, and Prove: Guiding Formal Theorem Provers with Informal Proofs

Anonymous submission

Abstract

1 The formalization of existing mathematical proofs is a notoriously difficult process.
2 Despite decades of research on automation and proof assistants, writing formal
3 proofs remains arduous and only accessible to a few experts. While previous studies
4 to automate formalization focused on powerful search algorithms, no attempts were
5 made to take advantage of available informal proofs. In this work, we introduce
6 *Draft, Sketch, and Prove (DSP)*, a method that maps informal proofs to formal proof
7 sketches, and uses the sketches to guide an automated prover by directing its search
8 to easier sub-problems. We investigate two relevant setups where informal proofs
9 are either written by humans or generated by a language model. Our experiments
10 and ablation studies show that large language models are able to produce well-
11 structured formal sketches that follow the same reasoning steps as the informal
12 proofs. Guiding an automated prover with these sketches enhances its performance
from 20.9% to 39.3% on a collection of mathematical competition problems.



13 Figure 1: **Draft, Sketch, and Prove**. Starting with an informal statement, our framework yields a formal proof through a three-stage process: drafting informal proofs, mapping them into formal sketches, and proving the remaining conjectures. Concretely, an informal statement is a mathematical problem described in a mixture of natural and mathematical languages (e.g., formulae in \LaTeX). Then, we use a large language model to autoformalize each informal proof into a formal sketch, which is a skeleton of the formal proof with open conjectures left unproven (indicated by the `<proof>` blocks). The formal sketch mirrors the structure of the informal proof. Finally, the open conjectures/gaps inside each formal sketch are proved by an off-the-shelf prover.

14 1 Introduction

15 Formal proof automation is a challenging task that has been the focus of increased attention in recent
16 years (Bansal et al., 2019b; Polu & Sutskever, 2020; Lample et al., 2022; Jiang et al., 2022; Wu et al.,
17 2022). However, deep learning approaches have not been as successful as in other domains, mainly
18 because of the scarcity of formal data. To address the scarcity of formal proofs, previous studies
19 have proposed to use synthetic data (Wu et al., 2021), self-supervision (Polu & Sutskever, 2020;
20 Han et al., 2022), or reinforcement learning (Bansal et al., 2019a; Polu et al., 2022) to synthesize

21 additional formal training data. Although these methods alleviate the data insufficiency to some
22 degree, none are able to capitalize on the bulk of human-written mathematical proofs.

23 We give a schematic diagram of the *DSP* method in Figure 1 and describe it in Section 2. Recent
24 work (Wu et al., 2022) demonstrates the feasibility of automatically translating informal statements
25 into formal ones with large language models. *DSP* goes beyond and leverages large language models
26 to generate *formal proof sketches* (Wiedijk, 2003) from *informal proofs*. Proof sketches consist of
27 high-level reasoning steps that can be interpreted by formal systems such as interactive theorem
28 provers. They differ from complete formal proofs in that they contain sequences of intermediate
29 conjectures without justification. An example of informal proof with its corresponding formal proof
30 sketch is provided in Appendix A. In the last step of *DSP*, we elaborate the formal proof sketch into
31 a full formal proof using an automated prover to prove all intermediate conjectures.

32 We perform experiments to generate formal proofs of problems from the miniF2F dataset (Zheng
33 et al., 2022) and show that a large portion of theorems can be proved automatically with this method.
34 We investigate two settings where the informal proofs are either written by humans or drafted by
35 a large language model trained on mathematical text. These two settings correspond to situations
36 frequently occurring during the formalization of existing theories, where informal proofs are usually
37 available, but sometimes left as exercises to the reader or missing due to space limits in the margin.

38 **Contributions:**

- 39 • We introduce a novel approach to leverage informal proofs to guide automated provers with
40 formal proof sketches.
- 41 • To evaluate our approach, we build a dataset of manually curated informal statements and
42 informal proofs aligned with formal statements in the miniF2F dataset (Zheng et al., 2022).
- 43 • We increase the proportion of problems solved by an automated prover on miniF2F from
44 20.9% to 37.7%, given language-model-generated informal proofs, and up to 39.3% when
45 proofs are written by humans.

46 **2 Method**

47 In this section, we describe our *Draft, Sketch, and Prove (DSP)* method for formal proof automation,
48 which leverages informal proofs to guide automated formal theorem provers with proof sketches. We
49 assume that each problem comes with an informal statement and a formal statement describing the
50 problem. Our pipeline consists of three stages (depicted in Figure 1), which we present below.

51 **2.1 Drafting informal proofs**

52 The initial phase of the *DSP* method consists in finding an informal proof for a problem according to
53 its description in natural mathematical language. The resulting informal proof is seen as a *draft* for the
54 subsequent steps. In mathematical textbooks, proofs of theorems are in general provided, but are some-
55 times missing or incomplete. Therefore, we consider two settings corresponding to the presence or ab-
56 sence of the informal proofs. In the first, we assume that a “ground-truth” informal proof (i.e., written
57 by a human) is available, which is a typical scenario in the formalization of existing mathematical the-
58 ories. In the second setting, we make a more general assumption that the ground-truth informal proof
59 is not given, and draft proof candidates with a large language model trained on informal mathematical
60 data. The language model removes the dependence on human proofs and can produce multiple alterna-
61 tive solutions for every problem. Although there is no easy way to automatically verify the correctness
62 of these proofs, the informal proof only needs to be *useful* for producing a sketch in the next stage.

63 **2.2 Mapping informal proofs into formal sketches**

64 A formal proof sketch encodes the structure of a solution and leaves out low-level details (Wiedijk,
65 2003). Intuitively, it is a partial proof that outlines high-level conjecture statements. A concrete
66 example of a proof sketch is shown in Figure 2. Although informal proofs often leave aside low-level
67 details, (e.g., by stating their triviality), these details cannot be discharged in a formal proof, making
68 straightforward informal-to-formal proof translation difficult. Instead, we propose to map informal
69 proofs to formal proof sketches that share the same high-level structures. The low-level details

70 missing from a proof sketch can later be filled by an automated prover. Since large informal-formal
71 parallel corpora do not exist, standard machine translation methods are unsuitable for this task.
72 Rather, we use the few-shot learning abilities of a large language model. Specifically, we prompt the
73 model with a few example pairs containing *informal proofs* and their corresponding *formal sketches*,
74 followed by an informal proof yet to be translated. We then let the model generate the subsequent
75 tokens to obtain the desired formal sketch. We refer to this model as an autoformalizer.

76 2.3 Proving open conjectures in the sketches

77 As the last part of the process, we rely on off-the-shelf automated provers to fill in the missing
78 low-level details in the proof sketches, where an “automated prover” refers to a system capable
79 of producing formally verifiable proofs. Our framework is agnostic to the specific choice of the
80 automated prover: it can be symbolic provers such as heuristic tactics or proof automation tools,
81 neural-network-based provers, or even hybrid approaches. If the automated prover successfully closes
82 all the gaps in the proof sketch, it returns the final formal proof which can be checked against the
83 problem’s specification. If the automated prover fails (e.g., it exceeds the allocated time limit), we
84 consider the evaluation to be unsuccessful.

85 3 Experiments

86 3.1 Dataset and evaluation

87 We evaluate our method on the miniF2F dataset (Zheng et al., 2022). The dataset contains the *formal*
88 *statements* of 488 problems from high-school mathematical competitions, written in three formal
89 languages: Lean, HOL-Light, and Isabelle. They are split into a valid set and a test set, composed of
90 244 problems each. In this work, we choose to experiment with Isabelle.

91 The miniF2F dataset is comprised of problems from three source categories: (1) 260 problems sam-
92 pled from the MATH dataset (Hendrycks et al., 2021); (2) 160 problems from actual high-school math-
93 ematical competitions (AMC, AIME, and IMO); (3) 68 crafted problems at the same difficulty level as
94 (2). We employ three different methods to obtain informal statements and proofs from these sources.
95 For source (1), we access the informal statements and proofs from the MATH dataset; for (2), we
96 retrieve their informal statements and proofs from the AOPS website ¹; and for (3), we manually write
97 down their informal statements and proofs. Thus we gather a parallel set of 488 informal statements,
98 informal proofs, and formal statements. This dataset provides the informal statements and proofs for
99 our experiment in the human-as-informal-proof-writer setting and will be released upon publication.

100 Our task is to generate formal proofs for problems as they are formally stated in miniF2F. We consider
101 a proof valid if and only if it (a) does not contain “cheating” keywords (`sorry` and `oops`) that exit
102 a proof without completing it, and (b) Isabelle is able to verify the corresponding formal statement
103 with the proof. We use the Portal-to-ISabelle API by Jiang et al. (2021) to interact with Isabelle.

104 3.2 Baselines

105 **Sledgehammer** As a baseline, we attempt to prove the formal statement directly with Sledgehammer,
106 a popular proof automation tool in Isabelle. We use the default Sledgehammer configuration in
107 Isabelle2021, including a 120-second timeout and the five automated theorem provers (Z3, CVC4,
108 SPASS, Vampire, E). Appendix D gives a more thorough introduction to Sledgehammer.

109 **Sledgehammer + heuristics** Occasionally, Sledgehammer may fail without trying simple yet effective
110 tactics. As a second, stronger baseline, we create an automated prover that tries 11 common tactics
111 (`auto`, `simp`, `blast`, `fastforce`, `force`, `eval`, `presburger`, `sos`, `arith`, `linarith`,
112 `auto simp: field_simps`) for high-school level algebra and number theory problems. If every
113 attempted tactic fails, or times out after 10 seconds, it falls back to Sledgehammer.

114 **Language models for proof search** Finally, we include baselines representative of state-of-the-
115 art neural theorem proving in Isabelle, specifically Thor (Jiang et al., 2022) and Thor with expert
116 iteration on autoformalized data (Wu et al., 2022). GPT-f with expert iteration (Polu et al., 2022) and
117 HyperTree Proof Search (HTPS) (Lample et al., 2022) can solve 36.6% and 41.0% of the problems

¹<https://artofproblemsolving.com/community>

Table 1: **Proving success rates on the miniF2F dataset with Isabelle** In the table are the success rates of four baselines, the DSP method with human and language model informal proofs, as well as two ablation studies, on the validation and the test sets of miniF2F. The highest success rates on each set are highlighted in bold. The performance difference between ablation studies and DSP with human informal proofs are enclosed in brackets.

Success rate	miniF2F-valid	miniF2F-test
<i>Baselines</i>		
Sledgehammer	9.9%	10.4%
Sledgehammer + heuristics	18.0%	20.9%
Thor (Jiang et al., 2022)	28.3%	29.9%
Thor + expert iteration (Wu et al., 2022)	37.3%	35.2%
<i>Draft, Sketch, and Prove</i>		
Human informal proof	42.6%	39.3%
Codex informal proof	40.6%	35.3%
8B Minerva informal proof	40.6%	35.3%
62B Minerva informal proof	43.9%	37.7%

118 on miniF2F-test. However, they are not directly comparable to our method, as they rely on the Lean
 119 theorem prover, which greatly influences the performance due to the different tactics and automation.

120 3.3 Results

121 We cover our detailed experimental setup in Appendix E. In Table 1, we display the proportion of
 122 successful formal proofs found on the miniF2F dataset with the interactive theorem prover Isabelle.
 123 The results include the four baselines described in Subsection 3.2 and the *DSP* method with human-
 124 written proofs and model-generated proofs. From the table, we can see that the automated prover with
 125 11 additional heuristic tactics significantly increases the performance of Sledgehammer, boosting its
 126 success rate from 9.9% to 18.0% on the validation set of miniF2F and from 10.4% to 20.9% on the
 127 test set. The two baselines using language models and proof search (Thor and Thor + expert iteration)
 128 achieve success rates of 29.9% and 35.2% on the test set of miniF2F, respectively.

129 With informal proofs written by humans, the *DSP* method achieves success rates of 42.6% and
 130 39.3% on the validation and test sets of miniF2F. A total of 200 out of 488 problems can be proved
 131 in this way. The Codex model and the Minerva (8B) model give very similar results in solving
 132 problems on miniF2F: they both guide the automated prover to solve 40.6% and 35.3% of problems
 133 on the validation and the test sets respectively. This is corroborated by Lewkowycz et al. (2022)’s
 134 observation that these two models have comparable performances in solving mathematical problems.

135 When we switch to the Minerva (62B) model, the success rates rise up to 43.9% and 37.7% re-
 136 spectively. Compared to human-written informal proofs, its success rates are 1.3% higher on the
 137 validation set and 1.6% lower on the test set. In total, the Minerva (62B) model is able to solve 199
 138 problems on miniF2F, one fewer than with human proofs. The *DSP* method is effective in guiding
 139 the automated prover under both settings that we study: using either human informal proofs or
 140 language-model-generated informal proofs. *DSP* almost doubles the prover’s success rate and results
 141 in a new state-of-the-art performance on miniF2F with Isabelle. Moreover, the larger Minerva model
 142 is almost as helpful as a human in guiding the automated prover in solving problems.

143 4 Conclusion

144 In this paper, we introduced *Draft, Sketch, and Prove (DSP)*, a novel approach that takes advantage
 145 of informal proofs to synthesize formal proofs. We demonstrated its feasibility and effectiveness
 146 by reaching state-of-the-art performance on the miniF2F dataset with the Isabelle theorem prover.
 147 Central to our method are formal proof sketches that mirror the high-level reasoning structures of
 148 informal proofs. Our ablations showed that the ability to automatically convert informal proofs to
 149 proof sketches is critical to the success of *DSP*.

150 **References**

- 151 Kshitij Bansal, Sarah M. Loos, Markus N. Rabe, and Christian Szegedy. Learning to reason in large
152 theories without imitation. *CoRR*, abs/1905.10501, 2019a. URL [http://arxiv.org/abs/](http://arxiv.org/abs/1905.10501)
153 [1905.10501](http://arxiv.org/abs/1905.10501).
- 154 Kshitij Bansal, Sarah M. Loos, Markus N. Rabe, Christian Szegedy, and Stewart Wilcox. Holist: An
155 environment for machine learning of higher order logic theorem proving. In Kamalika Chaudhuri
156 and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine*
157 *Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings*
158 *of Machine Learning Research*, pp. 454–463. PMLR, 2019b. URL [http://proceedings.](http://proceedings.mlr.press/v97/bansal19a.html)
159 [mlr.press/v97/bansal19a.html](http://proceedings.mlr.press/v97/bansal19a.html).
- 160 Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Jean-Christophe Filliatre, Eduardo
161 Gimenez, Hugo Herbelin, Gerard Huet, Cesar Munoz, Chetan Murthy, et al. *The Coq proof*
162 *assistant reference manual: Version 6.1*. PhD thesis, Inria, 1997.
- 163 Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhari-
164 wal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agar-
165 wal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh,
166 Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler,
167 Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCand-
168 lish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot
169 learners. In Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan,
170 and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual*
171 *Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12,*
172 *2020, virtual*, 2020. URL [https://proceedings.neurips.cc/paper/2020/hash/](https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfc4967418bfb8ac142f64a-Abstract.html)
173 [1457c0d6bfc4967418bfb8ac142f64a-Abstract.html](https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfc4967418bfb8ac142f64a-Abstract.html).
- 174 Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison
175 Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger,
176 Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick
177 Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter,
178 Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis,
179 Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Arun
180 Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa,
181 Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder,
182 Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating
183 large language models trained on code. *ArXiv*, abs/2107.03374, 2021.
- 184 Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam
185 Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh,
186 Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam
187 Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James
188 Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Lev-
189 skaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin
190 Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret
191 Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick,
192 Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Er-
193 ica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang,
194 Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern,
195 Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language model-
196 ing with pathways. *CoRR*, abs/2204.02311, 2022. doi: 10.48550/arXiv.2204.02311. URL
197 <https://doi.org/10.48550/arXiv.2204.02311>.
- 198 Iddo Drori, Sarah Zhang, Reece Shuttleworth, Leonard Tang, Albert Lu, Elizabeth Ke, Kevin Liu,
199 Linda Chen, Sunny Tran, Newman Cheng, et al. A neural network solves, explains, and generates
200 university math problems by program synthesis and few-shot learning at human level. *Proceedings*
201 *of the National Academy of Sciences*, 119(32):e2123433119, 2022.
- 202 Thibault Gauthier, Cezary Kaliszyk, Josef Urban, Ramana Kumar, and Michael Norrish. Tactictoe:
203 learning to prove with tactics. *Journal of Automated Reasoning*, 65(2):257–286, 2021.

- 204 Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward W. Ayers, and Stanislas Polu. Proof artifact
205 co-training for theorem proving with language models. In *The Tenth International Conference on*
206 *Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.
207 URL <https://openreview.net/forum?id=rpXJc9j04U>.
- 208 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,
209 and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*,
210 2021.
- 211 Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text
212 degeneration. *arXiv preprint arXiv:1904.09751*, 2019.
- 213 Albert Q. Jiang, Wenda Li, Jesse Michael Han, and Yuhuai Wu. LISA: Language models of Isabelle
214 proofs. In *6th Conference on Artificial Intelligence and Theorem Proving*, 2021.
- 215 Albert Q. Jiang, Wenda Li, Szymon Tworkowski, Konrad Czechowski, Tomasz Odrzygózdz, Piotr
216 Milos, Yuhuai Wu, and Mateja Jamnik. Thor: Wielding hammers to integrate language models
217 and automated theorem provers. *CoRR*, abs/2205.10893, 2022. doi: 10.48550/arXiv.2205.10893.
218 URL <https://doi.org/10.48550/arXiv.2205.10893>.
- 219 Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International*
220 *Conference on Learning Representations*, 2020. URL [https://openreview.net/forum?](https://openreview.net/forum?id=SlEZYeHFDS)
221 [id=SlEZYeHFDS](https://openreview.net/forum?id=SlEZYeHFDS).
- 222 Guillaume Lample, Marie-Anne Lachaux, Thibaut Lavril, Xavier Martinet, Amaury Hayat, Gabriel
223 Ebner, Aurélien Rodriguez, and Timothée Lacroix. Hypertree proof search for neural theorem
224 proving. *CoRR*, abs/2205.11491, 2022. doi: 10.48550/arXiv.2205.11491. URL [https://doi.](https://doi.org/10.48550/arXiv.2205.11491)
225 [org/10.48550/arXiv.2205.11491](https://doi.org/10.48550/arXiv.2205.11491).
- 226 Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V.
227 Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam
228 Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with
229 language models. *CoRR*, abs/2206.14858, 2022. doi: 10.48550/arXiv.2206.14858. URL [https:](https://doi.org/10.48550/arXiv.2206.14858)
230 [//doi.org/10.48550/arXiv.2206.14858](https://doi.org/10.48550/arXiv.2206.14858).
- 231 Norman D. Megill and David A. Wheeler. *Metamath: A Computer Language*
232 *for Mathematical Proofs*. Lulu Press, Morrisville, North Carolina, 2019.
233 <http://us.metamath.org/downloads/metamath.pdf>.
- 234 Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The
235 lean theorem prover (system description). In *International Conference on Automated Deduction*,
236 pp. 378–388. Springer, 2015.
- 237 Lawrence C. Paulson. *Isabelle - A Generic Theorem Prover (with a contribution by T. Nipkow)*,
238 volume 828 of *Lecture Notes in Computer Science*. Springer, 1994. ISBN 3-540-58244-4. doi:
239 10.1007/BFb0030541. URL <https://doi.org/10.1007/BFb0030541>.
- 240 Lawrence C. Paulson. Three years of experience with sledgehammer, a practical link between
241 automatic and interactive theorem provers. In Renate A. Schmidt, Stephan Schulz, and Boris
242 Konev (eds.), *Proceedings of the 2nd Workshop on Practical Aspects of Automated Reasoning,*
243 *PAAR-2010, Edinburgh, Scotland, UK, July 14, 2010*, volume 9 of *EPiC Series in Computing*, pp.
244 1–10. EasyChair, 2010. doi: 10.29007/tnfd. URL <https://doi.org/10.29007/tnfd>.
- 245 Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving.
246 *CoRR*, abs/2009.03393, 2020. URL <https://arxiv.org/abs/2009.03393>.
- 247 Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya
248 Sutskever. Formal mathematics statement curriculum learning. *CoRR*, abs/2202.01344, 2022.
249 URL <https://arxiv.org/abs/2202.01344>.
- 250 David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez,
251 Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement
252 learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):
253 1140–1144, 2018.

- 254 Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks.
255 *Advances in neural information processing systems*, 27, 2014.
- 256 Donald Syme. *DECLARE: A prototype declarative proof system for higher order logic*. Citeseer,
257 1997.
- 258 Christian Szegedy. A promising path towards autoformalization and general artificial intelligence. In
259 Christoph Benzmüller and Bruce R. Miller (eds.), *Intelligent Computer Mathematics - 13th Interna-*
260 *tional Conference, CICM 2020, Bertinoro, Italy, July 26-31, 2020, Proceedings*, volume 12236 of
261 *Lecture Notes in Computer Science*, pp. 3–20. Springer, 2020. doi: 10.1007/978-3-030-53518-6_1.
262 URL https://doi.org/10.1007/978-3-030-53518-6_1.
- 263 Qingxiang Wang, Chad E. Brown, Cezary Kaliszzyk, and Josef Urban. Exploration of neural machine
264 translation in autoformalization of mathematics in mizar. In Jasmin Blanchette and Catalin Hritcu
265 (eds.), *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs*
266 *and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020*, pp. 85–98. ACM, 2020. doi:
267 10.1145/3372885.3373827. URL <https://doi.org/10.1145/3372885.3373827>.
- 268 Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho.
269 Naturalproofs: Mathematical theorem proving in natural language. In *Thirty-fifth Conference on*
270 *Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL
271 <https://openreview.net/forum?id=Jvxa8adr3iY>.
- 272 Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. Naturalprover:
273 Grounded mathematical proof generation with language models. *CoRR*, abs/2205.12910, 2022. doi:
274 10.48550/arXiv.2205.12910. URL <https://doi.org/10.48550/arXiv.2205.12910>.
- 275 Freek Wiedijk. Formal proof sketches. In Stefano Berardi, Mario Coppo, and Ferruccio Damiani
276 (eds.), *Types for Proofs and Programs, International Workshop, TYPES 2003, Torino, Italy,*
277 *April 30 - May 4, 2003, Revised Selected Papers*, volume 3085 of *Lecture Notes in Computer*
278 *Science*, pp. 378–393. Springer, 2003. doi: 10.1007/978-3-540-24849-1_24. URL https://doi.org/10.1007/978-3-540-24849-1_24.
- 280 Yuhuai Wu, Albert Jiang, Jimmy Ba, and Roger Baker Grosse. INT: An inequality benchmark
281 for evaluating generalization in theorem proving. In *International Conference on Learning*
282 *Representations*, 2021. URL <https://openreview.net/forum?id=O6LPudownQM>.
- 283 Yuhuai Wu, Albert Q. Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian
284 Szegedy. Autoformalization with large language models. *CoRR*, abs/2205.12615, 2022. doi:
285 10.48550/arXiv.2205.12615. URL <https://doi.org/10.48550/arXiv.2205.12615>.
- 286 Kaiyu Yang and Jia Deng. Learning to prove theorems via interacting with proof assistants. In
287 *International Conference on Machine Learning (ICML)*, 2019.
- 288 Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: a cross-system benchmark
289 for formal olympiad-level mathematics. In *The Tenth International Conference on Learning*
290 *Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL
291 <https://openreview.net/forum?id=9ZPegFuFTFv>.

292 **Appendix**

293 **A A proof sketch in Isabelle**

Informal Statement: Show that for any real number a , $10a \leq 28a^2 + 1$.

Informal Proof:
 It suffices to show $0 \leq 28a^2 - 10a + 1$. First, consider completing the square for $28a^2 - 10a$ and observe that $(a - \frac{5}{28})^2 = a^2 - \frac{10}{28}a + (5/28)^2$. Since $0 \leq (a - \frac{5}{28})^2$, we get $0 \leq a^2 - \frac{10}{28}a + (5/28)^2$. Multiplying by 28 and simplifying gives $0 \leq 28a^2 - 10a + (25/28)$. Since $25/28 < 1$, the result follows.

Formal Proof Sketch:

```

theorem algebra_binomnegdiscrineq_10alt28asqpl:
  fixes a :: real
  shows "10 * a ≤ 28 * a^2 + 1"
proof - (* it suffices to show 0 ≤ 28a^2 - 10a + 1 *)
  have c0: "0 ≤ 28a^2 - 10a + 1"
  proof - (* observe that (a - (5/28))^2 = a^2 - (10/28)a + (5/28)^2 *)
    have c1: "(a - (5/28))^2 = a^2 - 10/28a + (5/28)^2" <...>
      (* we get 0 ≤ a^2 - (10/28)a + (5/28)^2 *)
    have c2: "0 ≤ a^2 - 10/28a + (5/28)^2" using c1 <...>
      (* Multiplying by 28 and simplifying gives 0 ≤ 28a^2 - 10a + (25/28) *)
    have c3: "0 ≤ 28a^2 - 10a + 28((5/28)^2)" using c2 <...>
    have c4: "0 ≤ 28a^2 - 10a + 28((5/28)*(5/28))" using c3 <...>
    have c5: "0 ≤ 28a^2 - 10a + (25/28)" using c4 <...>
      (* Since 25/28 < 1, the result follows. *)
    show ?thesis using c5 <...>
  qed
  show ?thesis <...>
qed
    
```

Figure 2: **A proof sketch in Isabelle.** The problem “Show that for any real number a , $10a \leq 28a^2 + 1$ ” is given with an informal proof and an associated formal proof sketch. The sketch first rewrites the original statement (c0), which is proved through 5 intermediary conjectures (c1..c5). We use a special token (<...>) to indicate that the conjecture is “open” and should be tackled by an automated prover later. To facilitate the alignment between the informal and formal languages, we annotate the formal proof sketch examples with informal proof segments (shown in red), which are immediately followed by their formal counterparts.

294 **B Background and Related Work**

295 **Interactive theorem proving** Modern verification systems for mathematics are centered around
 296 *interactive theorem provers (ITPs)*, such as Isabelle (Paulson, 1994), Lean (Moura et al., 2015),
 297 Coq (Barras et al., 1997), or Metamath (Megill & Wheeler, 2019). ITPs embed the mathematical
 298 definitions and theorems onto a solid logical foundation (e.g., Higher-Order Logic, Dependent Type
 299 Theory) implemented by their kernels. Every theorem must be checked by the kernel to be recognized
 300 by the ITP. To be proved formally, a theorem is first stated in the ITP’s programming language, and
 301 iteratively simplified into simpler objectives (or subgoals), until it can be reduced to already proven
 302 facts. In this paper, we will refer to proofs verified by a formal theorem prover as *formal proofs*, and
 303 proofs written in “standard” mathematics (e.g. in $\mathcal{L}\text{TEX}$) as *informal proofs*.

304 **Machine learning for formal proof synthesis** Several approaches propose to combine machine
 305 learning with modern interactive theorem provers (Yang & Deng, 2019; Gauthier et al., 2021), and
 306 build upon the recent success of language models (Polu & Sutskever, 2020; Han et al., 2022; Polu
 307 et al., 2022; Jiang et al., 2022; Lample et al., 2022). These methods typically rely on sequence-to-
 308 sequence models (Sutskever et al., 2014) to generate the next step (or tactic) of a proof given the
 309 current proof state and perform search over the generated subgoals using powerful search methods

310 such as MCTS (Silver et al., 2018). Because search is computationally expensive, these language
311 models are relatively small (with fewer than 1 billion parameters). Our method contrasts with these
312 approaches in that we use a significantly reduced number of calls to the models, but also much larger
313 language models (with up to 175 billion parameters) that showcase outstanding few-shot learning
314 abilities (Brown et al., 2020).

315 **Machine learning for informal reasoning** Language models have also been used in the context
316 of purely informal mathematics (Lample & Charton, 2020; Hendrycks et al., 2021; Welleck et al.,
317 2021; Drori et al., 2022; Welleck et al., 2022). Nevertheless, Lewkowycz et al. (2022) note that for
318 quantitative question answering, models are prone to generate false positives: the model guesses
319 the right answer while providing an incorrect proof. These errors are hard to spot without human
320 inspection. Worryingly, the frequency of false positives increases with the difficulty of the problem.
321 Our method builds on these findings and translates informal proofs into formal proofs. Since ITPs
322 are logically grounded, once a formal proof is checked by them, we are guaranteed its correctness.

323 **Autoformalization** In a position paper, Szegedy (2020) argued for attaining formal mathematical
324 data from informal sources with neural networks. Wang et al. (2020) performed preliminary exper-
325 iments where the evaluation was limited to text-level similarities on synthetic datasets. Recently,
326 Wu et al. (2022) found that large language models (Chen et al., 2021; Chowdhery et al., 2022) are
327 capable of few-shot *statement autoformalization*. Namely, a small number of examples is enough to
328 learn to perform informal-to-formal translation of statements. In this paper, we investigate whether
329 these findings can generalize to *proof autoformalization*, i.e., whether large language models can be
330 used to translate informal proofs into formal ones.

331 C Conjectures and the declarative proof style

332 Interactive theorem provers such as Isabelle and Mizar use a *declarative* proof style (Syme, 1997), in
333 which a proof is interleaved with conjectures and their corresponding proofs. Syme (1997) stated
334 that the list of conjectures in a declarative proof should be analogous to a proof sketch found in a
335 mathematical textbook and sufficiently convincing for the reader. In practice, ITP users often prove a
336 theorem by writing down a list of conjectures (a “formal sketch”), then attempt to find a proof of
337 each conjecture (fill a “gap”) with an automated system.

338 D Sledgehammer

339 Sledgehammer (Paulson, 2010) is a powerful system that automates reasoning with the interactive
340 theorem prover Isabelle. It works by flattening the goals encoded in the higher-order logic used by
341 Isabelle/HOL into other logics (e.g., first-order logic) which can then be fed into automated theorem
342 provers such as E², CVC4³, Z3⁴, Vampire⁵, and SPASS⁶. If any of these automated theorem
343 provers succeeds in finding the proof in their own corresponding format, Sledgehammer reconstructs
344 the proof in Isabelle/HOL with certified provers (*metis*, *meson*, and *smt*), which is relatively
345 more interpretable by humans.

346 As a practical example of using Sledgehammer, one can declare a conjecture in Isabelle/HOL:
347 `have "4 dvd (a::nat) \implies 2 dvd a"` and call Sledgehammer immediately afterwards.
348 If Sledgehammer succeeds, it will return a proof step that proves the conjecture. In this example,
349 the step is by `(meson dvd.trans even_numeral)`, which uses the *meson* resolution prover
350 and two facts: that the division relation is transitive and that 4 is an even number. If Sledgehammer
351 does not find the proof or timeouts, it will report failure.

²<https://www.lehre.dhbw-stuttgart.de/ssschulz/E/E.html>

³<https://cvc4.github.io/index.html>

⁴<https://github.com/Z3Prover/z3>

⁵<https://vprover.github.io/>

⁶<https://www.spass-prover.org/download/index.html>

352 E Experimental Setup

353 **Drafting** When informal proofs are generated, we condition a large language model on informal
354 statements to sample 100 informal proofs per problem. Specifically, we use the Codex code-danvici-
355 002 model (Chen et al., 2021) through the OpenAI API, and the 8B and the 62B versions of the
356 Minerva model from Lewkowycz et al. (2022). We use greedy decoding for Codex and nucleus
357 sampling (Holtzman et al., 2019) with temperature $T = 0.6$ and $\text{top}_p = 0.95$ for Minerva models.

358 **Sketching** For sketching, we manually prepare 20 autoformalization examples of the format (*informal*
359 *statement, informal proof, formal statement, formal sketch*), to form a pool of high-quality demon-
360 strations. Of these 20 examples, 10 are of the *algebra* type and 10 are of the *number theory* type. All
361 examples are from the validation set of the miniF2F dataset and can be found in the supplementary
362 materials. The sketches contain in-line comments as in Figure 2. If the name of the problem gives
363 away its type (*algebra* or *number theory*), we only use examples of the corresponding type. We also
364 ensure that the sampled few-shot examples do not contain the problem being solved. The prompt
365 is composed of 3 uniformly randomly sampled example from the pool and the current problem’s
366 (*informal statement, informal proof, formal statement*). We use this prompt to query the same Codex
367 model to get the desired proof sketches. We use greedy decoding and a maximum of 2048 tokens
368 in the generated sequence. For all the experiments, we control the total number of queries made to
369 Codex per problem to be 100. This means 100 queries per human informal solution and one query
370 per language-model-generated solution .

371 **Proving** To prove the conjectures left open by the formal sketch, we use the Sledgehammer +
372 heuristics automated prover described in Subsection 3.2. We execute the automated prover on every
373 open conjecture in the sketch to synthesize a formal proof that can be verified by Isabelle.

374 F Analysis

375 F.1 Ablation studies

376 **Ablation of in-line comments** To facilitate the alignment between the informal proofs and the
377 formal proof sketches, we copy relevant segments of the informal proofs as in-line comments in
378 the sketches. In the manually constructed prompt examples, these comments are prefixed to the
379 corresponding Isabelle code blocks, as shown in Figure 2 (the text in red). We hypothesize that this
380 technique is beneficial for large language models to synthesize formal sketches. To validate this
381 hypothesis, we perform an ablation study by removing the in-line comments in the prompt examples
382 before running the experiment. The results are displayed in Table 1. We find that without in-line
383 comments, the success rates drop by 4.9% and 2.8% on the validation and test sets respectively. We
384 conclude that having in-line comments is helpful for generating formal proof sketches.

385 **Ablation of sketching** Having formal proof sketches as an intermediate data mode that are later
386 completed by automated provers is central to our method. To study the effect of this critical component,
387 we conduct an experiment without the sketching operation. Namely, we replace the formal proof
388 sketches with complete formal proofs in the prompt examples. As a result, the large language model
389 will also generate formal proofs instead of sketches. We directly verify whether these generated proofs
390 are valid. The results in this setup are presented in Table 1. The results reveal that without sketching,
391 the success rate on miniF2F decreases by 9.8% and 9.0% on the validation and test sets respectively.
392 The drastic performance difference indicates the essential role of proof sketches in our approach.

393 **Scaling properties of ablation studies** To understand the effect of the ablations on the *DSP*
394 method’s scaling properties, we vary the number of autoformalization attempts per problem and plot
395 the number of successful proofs found on the miniF2F dataset in Figure 3 (left). Three methods are
396 contrasted: the original *DSP* method with human informal proofs, the *DSP* method without in-line
397 comments, and the *DSP* method without sketching. It can be seen from the figure that with the original
398 *DSP* method, the performance reaches a plateau (no new proofs are found) after 70 autoformalization
399 attempts are made for each problem. For the ablation study with no in-line comments, the plateau is
400 reached much faster, after around 50 autoformalization attempts. This method solves 181 problems in
401 total. The ablation study without sketching can solve 154 problems on miniF2F. In comparison, with
402 human informal proofs, only 7 autoformalization attempts are required to reach this performance.

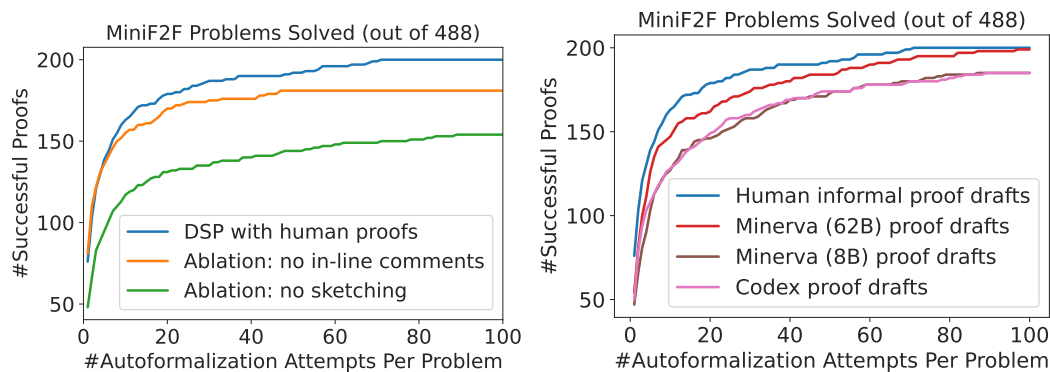


Figure 3: **Number of problems solved on miniF2F against the number of autoformalization attempts per problem** **Left:** The figure displays the experiments carried out with the full human informal proofs. The curves represent the full *DSP* method (blue), formal proof sketches without the in-line comments (orange), and without the formal proof sketches (green). **Right:** This figure compares the experimental results with informal proof drafts written by humans (blue), the 62B Minerva model (red), the 8B Minerva model (brown), and the Codex model (pink).

403 F.2 Language-model-generated proofs

404 Our experiments demonstrated that model-generated informal proofs from Minerva and Codex can
 405 help guide a formal theorem prover. In this section, we analyze the properties of these proofs further.
 406 Since the Minerva (62B) model gives the best overall performance on miniF2F, we focus on the
 407 informal proofs it produces in this section.

408 **Minerva helps solve one IMO problem** Interestingly, our approach manages to solve one problem
 409 from the International Mathematical Olympiad (`imo_1959_1`) with a Minerva-generated solution,
 410 but not with the human proof. For this problem, we present the successful Minerva-generated
 411 informal proof draft and the formal proof in Figure 4. We hypothesize that the reason behind this
 412 phenomenon is that human proofs might leave gaps between conjectures that are too difficult for
 413 automated provers to solve. On the other hand, the diversity in language model informal proofs
 414 makes some of them more amenable to automated provers. In Appendix G, we analyze the human
 415 and the Minerva informal proofs for this problem in greater detail.

416 **Manual evaluation of Minerva proofs** Next, we analyze the relationship between the validity of the
 417 formal proofs and the correctness of the informal proofs. For our analysis, we randomly sample
 418 50 Minerva proofs of different problems, which are then successfully converted to formal proofs. We
 419 then manually evaluate the correctness of these 50 informal proofs. Among them, 29 proofs (58%)
 420 are entirely correct, 16 are incorrect with a clearly identifiable incorrect step, and 5 “proofs” are
 421 nonsensical and simply rephrase the final conclusions of the problems.

422 Seeing that a total of $16 + 5 = 21$ incorrect informal proofs can lead to successful formal proofs,
 423 we study how they guide the automated formal prover despite having flaws themselves. The 21
 424 proofs divide into 2 cases: In the first case, we find 13 problems for which the informal proofs are
 425 mostly ignored, and the automated prover can find proofs by itself; In the other 8 problems, although
 426 the informal proofs are wrong, the autoformalizer manages to *correct* them, either by ignoring the
 427 erroneous steps or by stating their correct versions in the formal proof sketches. This suggests
 428 that the autoformalizer has some understanding of the mathematical statements and is not merely
 429 translating them from an informal language to a formal language. It is robust to slight noises in its
 430 input. In Appendix H, we present 3 case studies comparing the human and Minerva informal proofs.
 431 Particularly, Figure 7 shows a completely correct example and one example of each pathological case.

432 Is there a way to detect which Minerva proofs are correct, without human evaluation? For a
 433 preliminary investigation, we filter out all the problems that can be solved directly with the automated
 434 prover from the 50 and are left with 27 informal proofs. Of these 27, 21 are completely correct, 6
 435 still contain small errors, but none are nonsensical. With this simple filter, we achieve a precision of
 436 77.8% and a recall of 72.4% in identifying correct Minerva informal proofs.

437 **Scaling properties of human and Minerva proofs** To understand the influence of different
 438 informal proof sources on the scaling properties of *DSP*, we plot the number of successful proofs found
 439 on miniF2F against the number of autoformalization attempts per problem in Figure 3 (right). Note
 440 that for each problem, we have 1 informal proof by a human and 100 informal proof drafts by each
 441 language model. The one human proof is used 100 times for formal proof sketch generation, while
 442 each language model proof draft is used only once. We notice that the 62B Minerva model results
 443 in more successful proofs than the smaller (8B) Minerva model and the Codex model consistently
 444 for any number of attempts. The 8B Minerva model and the Codex model behave similarly, both
 445 finding 185 proofs. Human informal proofs yield better results than Minerva (62B) for 1 – 100
 446 autoformalization attempts. However, the difference is small (1 problem) when 100 are made.

447 G A proof to an international mathematical olympiad problem

448 With the Minerva-generated solutions, a proof to the problem `imo_1959_p1` is discovered. This is
 449 the first problem of the first ever International Mathematical Olympiad (IMO). The informal problem
 450 statement, Minerva-generated informal solution, and DSP’s formal proof are shown in Figure 4.

451 In Figure 4, we can see that the autoformalizer in DSP (a large language model), copies over parts of
 452 the informal proof generated by Minerva as in-line comments to precede their corresponding formal
 453 proof blocks. The formal proof does not use the first sentence of the informal proof solution as it
 454 is already identical to the formal statement. We also notice that the large language model selects
 455 relevant premises after writing down the conjectures (the steps starting with `using`) despite not
 456 every premise is strictly needed.

457 The formal proof creates 5 conjectures (4 `have` statements and 1 `show` statement) which are all
 458 subsequently proved by our automated theorem prover. The step to prove the statement `have "gcd`
 459 `(21*n + 4) (14*n + 3) = 1"` involves 2 verified low-level provers `smt` and `z3` and 10
 460 lemmas/facts from outside the scope of the language model. It is highly unlikely that either the large
 461 language model or the automated theorem prover can finish this proof on its own.

462 **Unsuccessful human-written proof.** In contrast, the human-written informal proof of this IMO
 463 problem did not lead to a successful formal proof. The human-written proof is:

464 Denoting the greatest common divisor of a, b as (a, b) , we use the Euclidean algorithm:

$$(21n + 4, 14n + 3) = (7n + 1, 14n + 3) = (7n + 1, 1) = 1$$

465 It follows that $\frac{21n+4}{14n+3}$ is irreducible. Q.E.D.

466 A key difference between the Minerva proof and the human proof is the way that invoking the
 467 Euclidean algorithm is described. The Minerva proof explicitly writes out the results of the Euclidean
 468 algorithm (e.g. $21n + 4 = 1 \cdot (14n + 3) + 7n + 1$), which are translated into the sketch (`c1` in
 469 Figure 4). The human proof introduces new notation to express the results indirectly in terms of
 470 greatest common divisors, which ends up being less suitable for sketching. For example, below is
 471 a sketch generated with the human proof, which has a conjecture that is semantically incorrect and
 472 hence cannot be closed by the automated prover:

```
473 theorem
474   fixes n :: nat
475   shows "gcd (21*n + 4) (14*n + 3) = 1"
476 proof -
477   have "(21*n + 4, 14*n + 3) = (7*n + 1, 14*n + 3)"
478     ATP (* ← UNSUCCESSFUL *)
479   also have "... = (7*n + 1, 1)"
480     ATP
481   finally show ?thesis
482     ATP
483 qed
```

484 H More case analyses of human and Minerva informal proofs

Statement: Prove that the fraction $\frac{21n+4}{14n+3}$ is irreducible for every natural number n .

Informal Proof (Minerva 62B):

We must show that $\gcd(21n + 4, 14n + 3) = 1$.
The Euclidean algorithm gives

$$\begin{aligned} 21n + 4 &= 1 \cdot (14n + 3) + 7n + 1 \\ 14n + 3 &= 2 \cdot (7n + 1) + 1. \end{aligned}$$

Since $\gcd(7n + 1, 1) = 1$, we have $\gcd(21n + 4, 14n + 3) = 1$.

Formal Proof:

```

theorem imo_1959_p1:
  fixes n :: nat
  shows "gcd (21*n + 4) (14*n + 3) = 1"
proof -
  (* The Euclidean algorithm gives
  21n+4=1\cdot(14n+3)+7n+1
  14n+3=2\cdot(7n+1)+1. *)
  have c0: "21*n + 4 = 1*(14*n + 3) + 7*n + 1"
    by auto [ATP]
  have c1: "14*n + 3 = 2*(7*n + 1) + 1" using c0
    by auto [ATP]

  (* Since \gcd(7n+1,1)=1, we have \gcd(21n+4,14n+3)=1. *)
  then have "gcd (7*n + 1) 1 = 1"
    using c1
    by auto [ATP]
  then have "gcd (21*n + 4) (14*n + 3) = 1"
    using c1
    by (smt (z3) BitM_plus_one ab_semigroup_add_class.add_ac(1)
    add.assoc c0 gcd.commute gcd_add2 gcd_add_mult mult_numeral_1
    numeral_One numeral_eq_Suc numerals(1) semiring_norm(3)) [ATP]
  then
  show ?thesis
    using c1
    by blast [ATP]
qed

```

Figure 4: **IMO proof guided by a Minerva informal proof** An informal proof of the International Math Olympiad problem `imo_1959_p1` generated by Minerva that led to a successful formal proof. The steps suffixed by a `[ATP]` identifier are generated by an automated theorem prover and the rest are by the DSP autoformalizer.

Statement: Given that $f + 3z = 11$ and $3(f - 1) - 5z = -68$, show that $f = -10$ and $z = 7$.

Informal Proof (Minerva 62B):

We can solve for f in the first equation, obtaining $f = 11 - 3z$.
 Then we can substitute this expression for f into the second equation,
 obtaining $3(11 - 3z - 1) - 5z = -68$.
 Solving for z , we obtain $z = 7$.
 Then, we can substitute this value of z into the expression for f , obtaining $f = 11 - 3 \cdot 7 = -10$.

Formal Proof:

```

theorem algebra_2varlineareq_fp3zeq11_3tfm1m5zeqn68_feqn10_zeq7:
  fixes f z :: complex
  assumes h0: "f + 3*z = 11"
    and h1: "3*(f - 1) - 5*z = -68"
  shows "f = -10 &and z = 7"
proof -
  (* We can solve for f in the first equation, obtaining f = 11 - 3z. *)
  have c0: "f = 11 - 3*z"
    using h0
    by (auto simp: field_simps) [ATP]
  (* Then we can substitute this expression for f into the second equation,
    obtaining 3(11 - 3z - 1) - 5z = -68. *)
  have c1: "3*(11 - 3*z - 1) - 5*z = -68"
    using h1 c0
    by auto [ATP]
  (* Solving for z, we obtain z = 7. *)
  have c2: "z = 7"
    using c1
    by auto [ATP]
  (* Then, we can substitute this value of z into the expression for f,
    obtaining f = 11 - 3 \cdot 7 = -10. *)
  have "f = 11 - 3*7"
    using c0 c2
    by auto [ATP]
  then have c3: "f = -10"
    by auto [ATP]
  show ?thesis
    using c2 c3
    by auto [ATP]
qed

```

Figure 5: **Algebra example with Minerva informal proof.** An informal proof generated by Minerva that led to a successful formal proof. The autoformalizer generated a proof sketch containing all lines of the formal proof except for those with *[ATP]*. The sketch is structured according to the informal proof, containing five intermediate conjectures based on the informal proof. The autoformalizer generated in-line comments in the proof sketch (shown in red), which correctly identified an alignment between the formal and informal proofs.

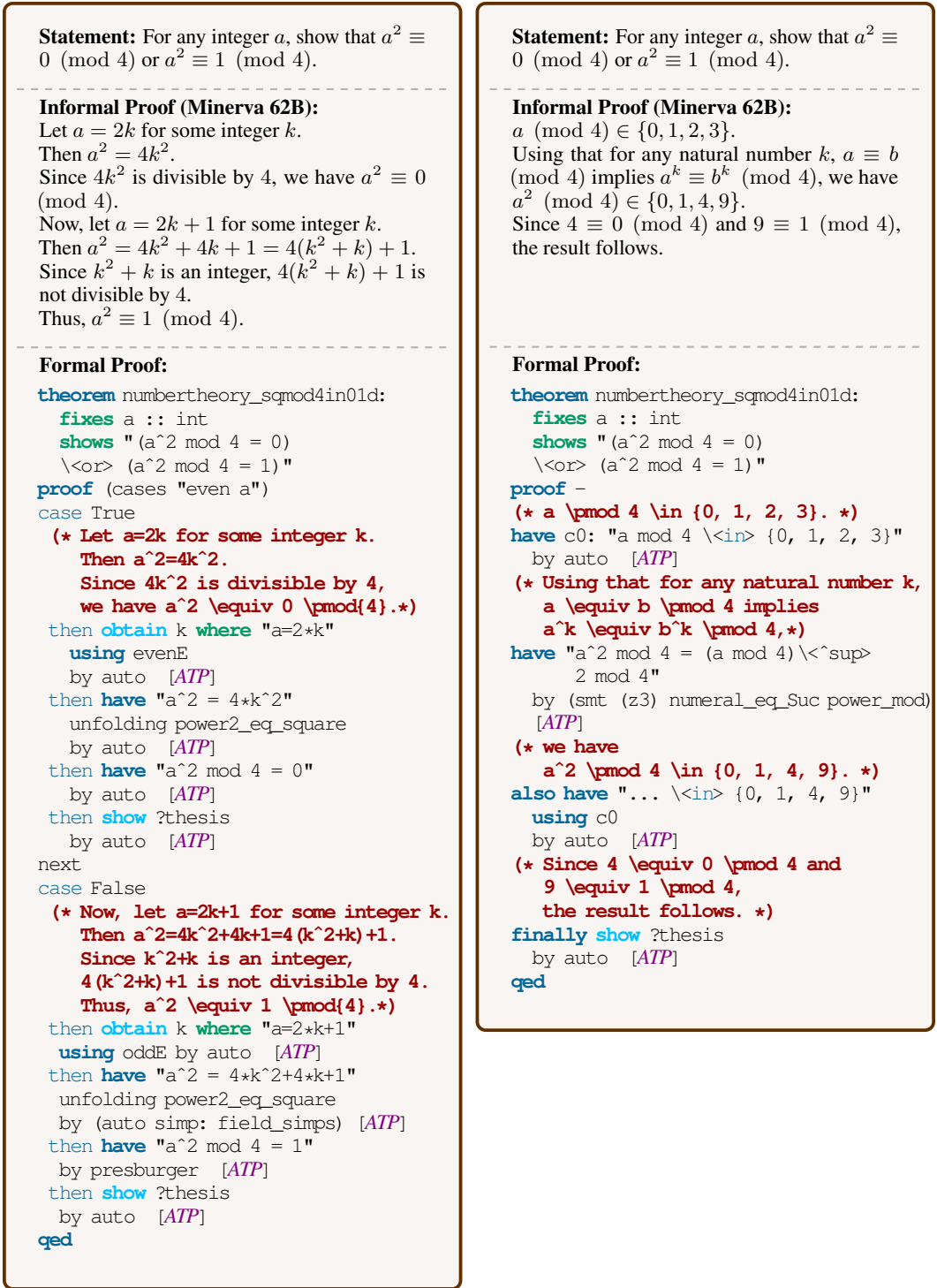


Figure 6: **Alternative proofs: Minerva (left) and human-written (right).** In both proofs, the formal sketch is structured based on the informal proof. The Minerva informal proof and its sketch break the proof into even and odd cases. These cases are not explicitly stated in the informal proof, and the formal sketch makes them explicit (cases “even a”). Each case has three conjectures that are directly based on the informal proof. The sketch excludes parts that are not needed in the formal proof, for instance “Since $k^2 + k$ is an integer, $4(k^2 + k) + 1$ is not divisible by 4”. The human proof uses a different strategy than the Minerva proof, based on the facts that $a \pmod{4} \in \{0, 1, 2, 3\}$ and $a^2 \pmod{4} \in \{0, 1, 4, 9\}$. The sketch uses these as conjectures, synthesizes an intermediate step not in the informal proof, and excludes the last step of the informal proof.

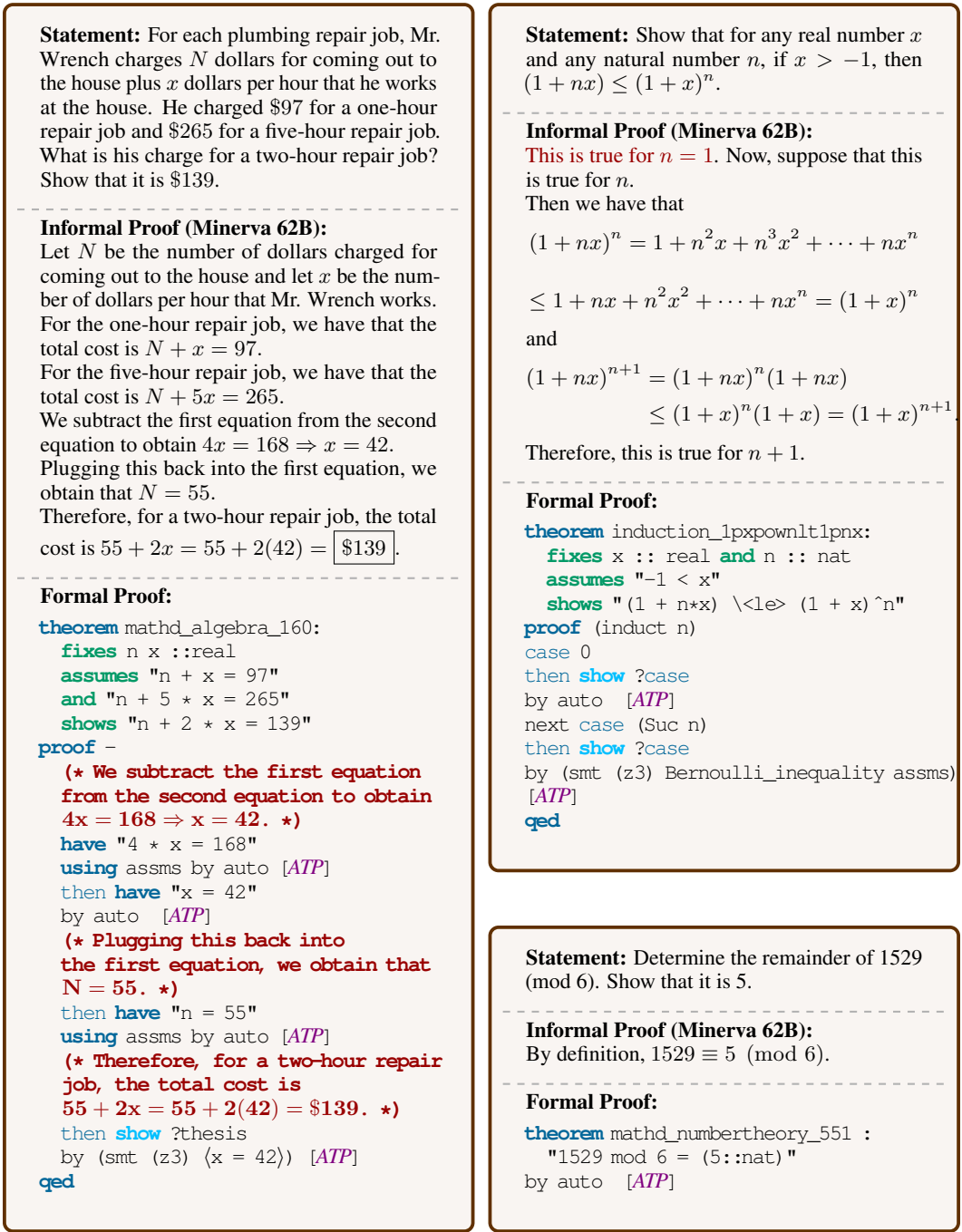


Figure 7: **Three Types of Minerva proofs: correct proof (left), incorrect proof (right top), nonsensical proof (right bottom)** In the correct Minerva proof, the formal sketch is structured based on the informal proof and steps are well-aligned. In the incorrect Minerva proof, the step "This is true for $n = 1$ " is corrected by Codex in the formal sketch to "case 0" which starts the base case with $n = 0$ since natural numbers include 0. This is an explicit correction made by Codex and makes a slightly incorrect Minerva proof formalized successfully. Lastly, the meaningless proof contains only a single statement without any calculation or justification. However, Codex also chooses to directly show the statement without any calculation. This suggests that the problem itself could be considered simple by Codex.