Autoformalization for Neural Theorem Proving

Yuhuai Wu[1], Albert Jiang[2], Wenda Li[2], Markus N. Rabe[1], Charles Staats[1],
Mateja Jamnik[2], and Christian Szegedy[1]

[1]Google Research
[2]University of Cambridge

1 Introduction

In this work, we demonstrate the feasibility and usefulness of autoformalization in the context of the newly introduced MiniF2F [10] benchmark. We use large language models to translate several thousands of informal problems into Isabelle and use them to improve our neural theorem prover. We find that transformer-based [7] language models trained on a large amount of web data are capable of formalizing mathematical competition problem statements with a relatively high success rate and the resulting statements can be used for creating new correct proofs that can be used for fine-tuning a neural theorem prover for improved proof automation. Using this methodology, we achieve a new state of the art on the MiniF2F benchmark.

2 Autoformalization using Large Language Models

Inspired by the success of large language models (LLMs) for synthesizing computer code by co-training on both natural language and code on web-scale data, we explore the capabilities of large language models (LLMs) that were trained on a large amount web data to turn natural language mathematics into formalized theorems (Isabelle theorems in this case). This is essentially a machine translation task [8] in which the input language is English and output language is formal code used by the interactive proof assistant Isabelle [9].

In particular, we exploit the impressive few-shot capability of LLMs by providing a few examples of the translations which improves the quality of our translation. We ran our initial experiments with using Codex and prompted the language model for the task of formalizing the informal statements. Here are two examples of automatically formalized theorems, with prompts provided in the Appendix.

Natural language version: *"Prove that there is no function f from the set of non-negative integers into itself such that $f(f(n)) = n + 1987$ for every n."* Translate the natural language version to an Isabelle version:

```
theorem
  fixes f :: "nat ⇒ nat"
  assumes "∀ n. f (f n) = n + 1987"
  shows False
```

Natural Language version: ”*When all the girls at Madeline’s school line up in rows of eight, there are seven left over. If instead they line up in rows of four, how many are left over? The final answer is 3.*”
Translate the natural language version to an Isabelle version:

```
theorem
  fixes n::nat
  assumes "n mod 8 = 7"
  shows "n mod 4 = 3"
```

Remarkably, we see in both examples, Codex was able to translate the natural language statement into Isabelle formal theorems perfectly. In the first example, the model can understand what it means by the phrase “to itself”, and correctly formalize the domain of function: f :: ”$nat \Rightarrow nat$”. The second example is even more remarkable. First of all, a formal translation of a grade school math problem should not ever exist in the pre-training corpus, as this type of mathematics is not of interest to formal mathematicians. Second, the examples in the prompt we provide also are not of this type of problem. It is hence remarkable that the model is capable of extrapolating to this type of statement – a true extrapolation. This shows a great promise of using LLMs for doing auto-formalization.

3 Autoformalization Improves Neural Theorem Proving

To study the usefulness of the formalized statements, we explore if one can improve neural theorem provers by training the model on automatically translated theorems. In particular, we study auto-formalization on a constrained setting – mathematical competition problems, where it has little requirement in formalizing the definitions and background theory.

For our neural theorem prover, we use a recently introduced theorem prover LISA [4] that proves Isabelle theorems by language modeling the best action conditioned on the current proof state. The input of the transformer-based neural network is the proof state and the output is the tactic application to be applied. This network is trained on existing human proofs. At inference time, a best-first search is performed using the neural network as an action generator.

Table 1: Proof rates on MiniF2F Benchmark

Model	valid	test
PACT [2]	23.9%	24.6%
FMSCL [5]	33.6%	29.6%
LISA [4]	28.3%	29.9%
LISA + AF	**36.1%**	**34.0%**

We use Codex [1] auto-formalize 3908 mathematical problems belonging to category `algebra`, `intermediate algebra`, and `number theory` from the training set of MATH [3]. Out of them, 3363 of the auto-formalized theorems are syntactically correct. We then use our neural prover trained on Isabelle corpus (AFP and Isabelle Standard library) to prove these theorems, and 23.3% of them can be proven. This gives us 782 new provably verified theorems along with their proofs for us to train our neural prover further. This form of training on one’s own generated data is known as expert iteration, and was already used in prior works [6, 5]. However, unlike in

Polu et. al. [5], where one perform expert iteration on a set of problems manually translated by human, we here use LLMs to auto-formalize the theorems.

After one epoch of training on the proofs of 782 theorems, we evaluated the neural prover on miniF2F [10], a recently introduced benchmark containing 488 mathematical competition statements manually formalized by humans. Some of those problems come from the valid and test set of MATH, and others come from previous International Mathematical Olympiad competitions or AoPS[1].

The results are shown in Table 1. LISA refers to the model before we trained on the autoformalized dataset, and LISA + AF refers to the model after one epoch of training on the 782 theorems. We see that by simply training on one epoch of the proved auto-formalized theorems, we can achieve a significant improvement in proof rate (from 28.3% to 36.1% on miniF2F-valid), and a new state-of-the-art performance on this benchmark.

4 Conclusion

For the first time, we have demonstrated that autoformalization is indeed feasible at least for high school mathematics competition problems and the translated results are useful for improving the performance of neural theorem provers.

However, our method is not capable of creating whole theories or autoformalization of facts that need to rely on libraries the language model has not been trained on. Full blown autoformalization of mathematical text will require new methods, especially proper training methodologies and utilizing newly introduced code by retrieval augmented language modeling.

References

[1] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

[2] Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward Ayers, and Stanislas Polu. Proof artifact co-training for theorem proving with language models. In *International Conference on Learning Representations*, 2022.

[3] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *CoRR*, abs/2103.03874, 2021.

[4] Albert Qiaochu Jiang, Wenda Li, Jesse Michael Han, and Yuhuai Wu. Lisa: Language models of isabelle proofs. *6th Conference on Artificial Intelligence and Theorem Proving*, 2021.

[5] Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning, 2022.

[1] https://artofproblemsolving.com/

[6] Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *CoRR*, abs/2009.03393, 2020.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[8] Qingxiang Wang, Chad Brown, Cezary Kaliszyk, and Josef Urban. Exploration of neural machine translation in autoformalization of mathematics in mizar. In *International Conference on Certified Programs and Proofs*, 2020.

[9] Makarius Wenzel, Lawrence C. Paulson, and Tobias Nipkow. The Isabelle framework. In Otmane Aït Mohamed, César A. Muñoz, and Sofiène Tahar, editors, *Theorem Proving in Higher Order Logics, 21st International Conference, TPHOLs 2008, Montreal, Canada, August 18-21, 2008. Proceedings*, volume 5170 of *Lecture Notes in Computer Science*, pages 33–38. Springer, 2008.

[10] Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*, 2021.

A Prompt

Natural language version: *"Let* $z = \frac{1+i}{\sqrt{2}}$, *find* $(\sum_{i=1}^{1} 2(z^{i^2})) \cdot (\sum_{i=1}^{1} 2(\frac{1}{z^{i^2}}))$. *The final answer is 36."*
Translate the natural language version to an Isabelle version:

```
theorem
  fixes z::complex
  assumes h0: "z = (Complex (1/sqrt 2) (1/sqrt 2))"
  shows "(∑k::nat=1..12. (z^(k^2)))
          * (∑ k::nat=1..12. 1/(z^(k^2))) =36"
```

Natural language version: *"Determine the value of ab if* $\log_8 a + \log_4 b^2 = 5$ *and* $\log_8 b + \log_4 a^2 = 7$. *The final answer is 512".* Translate the natural language version to an Isabelle version:

```
theorem
  fixes a b ::real
  assumes "(ln a) / (ln 8) + (ln (b^2)) / (ln 4) = 5"
          "(ln b) / (ln 8) + (ln (a^2)) / (ln 4) = 7"
  shows "a * b = 512"
```