GOEDEL-PROVER-V2: SCALING FORMAL THEOREM PROVING WITH SCAFFOLDED DATA SYNTHESIS AND SELF-CORRECTION

Yong Lin Shange Tang Bohan Lyu Ziran Yang Jui-Hui Chung
Haoyu Zhao Lai Jiang Yihan Geng Jiawei Ge Jingruo Sun
Jiayun Wu Jiri Gesi Ximing Lu David Acuna Kaiyu Yang
Hongzhou Lin Yejin Choi Danqi Chen Sanjeev Arora Chi Jin

Presenter: Yicheng Tao

September 22, 2025

Outline

Background

Automated Formal Theorem Proving

Provided with formalized statements of mathematical propositions, the model generates proofs which can be checked by proof assistants (e.g. Lean). Typically, there are two kinds of provers:

- Stepwise Prover: The prover receives feedbacks from verifier and produce proof step by step.
- Whole-Proof Generator: The prover generates the whole proof given the formal statement without interacting with the environment.

Featured Related Works

- **DeepSeek-Prover-V2**: Use 671B model for problem decomposition and 7B model for proving small lemmas.
- **Kimina-Prover**: Chain-of-thought Reasoning.

Challenges

Data Scarcity

The lack of high quality formal proof data is the main challenge for developing provers, usually addressed by data synthesis and in-house formalization.

Interaction with Verifier

Stepwise prover can obtain more information by interacting with the verifier, but has more engineering issues and is harder to train. The SOTA models are all whole-proof generator, which lacks interaction with the verifier.

Overview

- **Scaffolded Data Synthesis**
  - Formal statement corpus
- **Verifier-guided Self-Correction**
  - SFT and expert iteration
  - Multi-turn reinforcement learning
- **Model Averaging**
  - Average model parameters for diversity

Outline

Outline

Formalizer Training

Expert Iteration

1. Prompt Claude Sonnet 4 to generate 50K formalized statements with reasoning traces as initial dataset.
2. Perform SFT and use statements that pass both syntax check by verifier and semantic check by LLM for the next iteration.

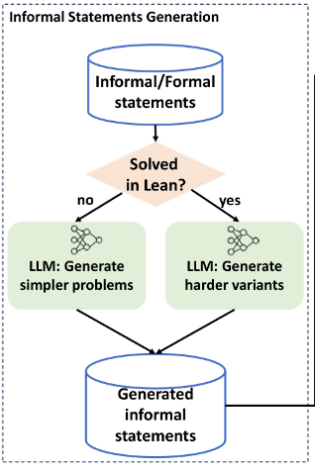
Feature

- Incorporating reasoning capability enables our formalizer to outperform previous models.

Name	Pass	Failed
Kimina-autoformalizer	161	139
Goedel-Formalizer-V2	228	72

Table 1: Comparison of different formalizers on 300 Omni-math problems.

Scaffolded Data Synthesis: Inforaml Statements Generation



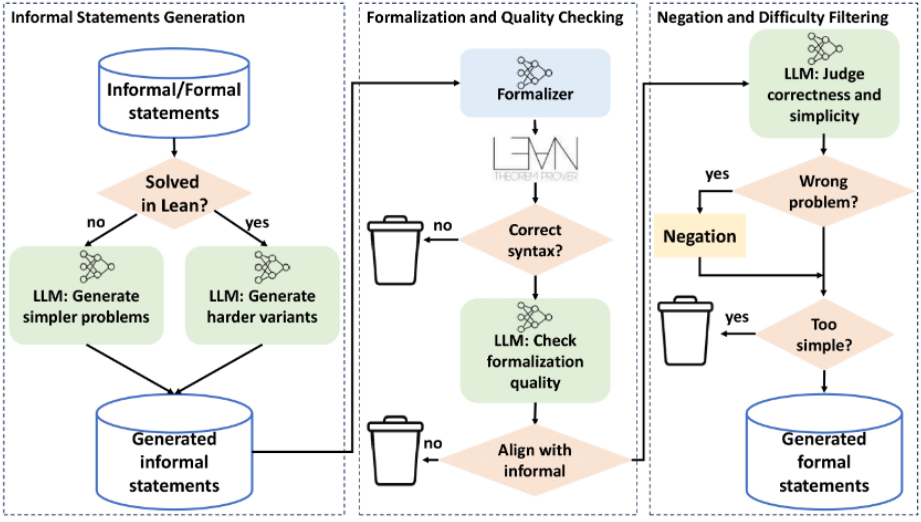
Formal-based synthesis

- Extract subgoals which represent easier subproblems from failed proofs.
- Introduce negated subgoals to prevent unprovable statements.
- These statements are used to augment training data of the next phase.

Informal-based synthesis

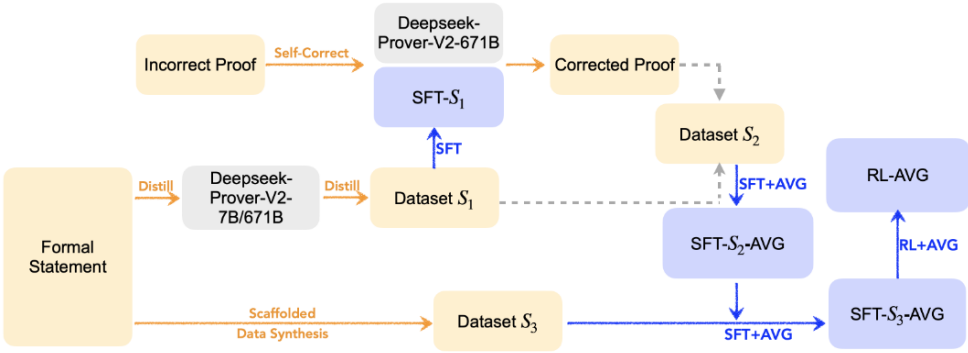
- If a given problem is (not) solved by the prover, then use LLM to generate harder (simpler) problems.
- Let the LLM generates natural language proofs first to enhance the quality of the generated problems.

Scaffolded Data Synthesis



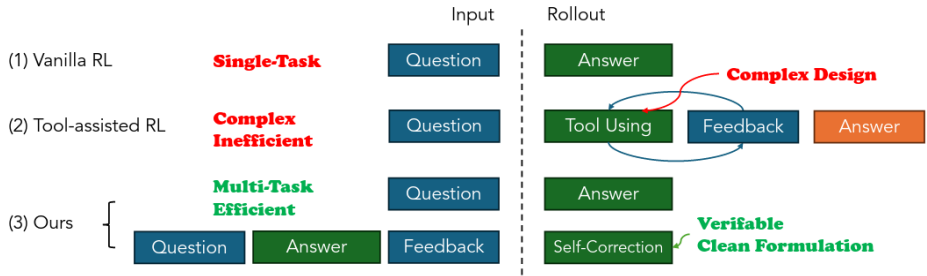
Outline

Supervised Fine-Tuning and Expert Iteration



- Use DeepSeek-Prover-V2 to initialize expert iteration.
- Enhance S_1 with self-correction traces forms S_2.

Reinforcement Learning



- Multi-task setup: 50% of the inputs are used for whole proof generation, and the remaining 50% for first-round self-correction.
- Hybrid GRPO: Remove group normalization and KL regularization. Incorporates clip-higher, overlong penalities and dynamic sampling.

Outline

Model Averaging

Motivation

- In latter stage of SFT and RL, the model's diversity decreases, reflected by the increase of pass@1 and decrease of pass@N.

Method

- Let the parameters of the base model be denoted as θ_0, and those of the fine-tuned model as θ. Use the combined model parameters defined as $(1-\alpha)\theta_0 + \alpha\theta$, where $\alpha \in (0, 1)$.
- Specifically, apply model averaging after completing SFT and use the averaged model as the starting point for RL. Once RL is completed, perform model averaging again and use this averaged model as the final model.

Outline

Benchmarks

- **MiniF2F**: 488 problems (244 validation and 244 test) in Lean from high-school level competitions including the AMC, AIME, and the International Mathematical Olympiad (IMO).
- **PutnamBench**: 644 problems focusing on college-level mathematics competition problems that are sourced from the William Lowell Putnam Mathematical Competition years 1962 - 2023.
- **MathOlympiadBench**: 360 problems sourced from Compfiles and IMOSLLean4 repository.

Main Results

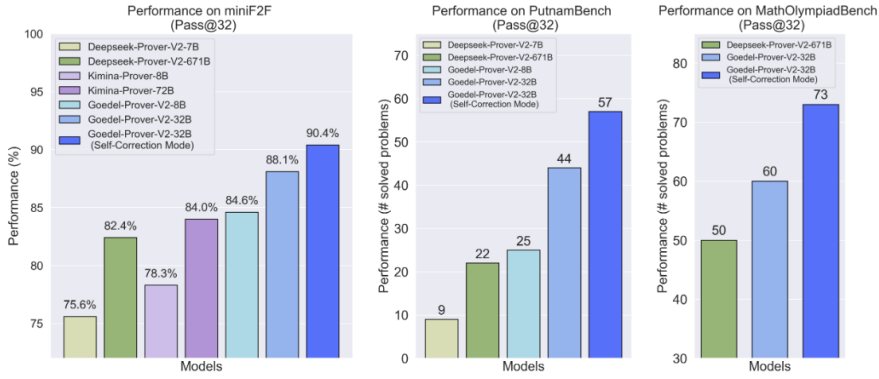


Figure 1: Performance of Goedel-Prover-V2 on different benchmarks under pass@32.

Main Results

- **High performance at modest scale**: 32B model outperforms DeepSeek-Prover-V2-671B. 8B variant nearly match or outperform Kimina-Prover-70B.
- **Efficacy of verifier-guided self-correction**: Adding self-correction provides a consistent gain of 2% on MiniF2F and 14 more solved in PutnamBench under pass@32.
- **Sample-efficient inference**: Goedel-Prover-V2 attains very high pass@N with minimal inference overhead (N=32 or 64).

Scaling Analysis

Model	32	64	128	256	512	1024	2048	4096	8192
32B (self-correction mode)	90.4	91.4	91.9	92.3	92.4	92.6	–	–	–
32B	88.1	89.8	90.5	91.0	91.6	91.8	92.0	92.2	92.2
8B (self-correction mode)	86.7	86.9	87.4	88.0	88.5	89.3	–	–	–
8B	84.6	86.1	86.5	87.0	87.4	87.9	88.5	89.3	90.2

Table 4: The performance (%) of Goedel-Prover-V2 on MiniF2F across different compute budget.

- These results indicate that Goedel-Prover-V2 efficiently internalizes reasoning during training, requiring fewer inference samples to achieve comparable or superior accuracy.
- Verifier-guided self-correction consistently improve the performance of the prover, indicating the value of combining CoT reasoning and error correction in formal theorem proving.

Analysis of Self-Correction

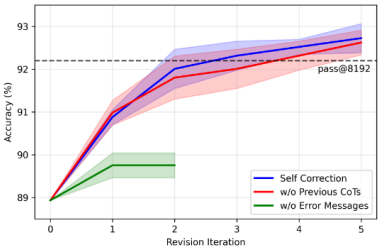


Figure 7: Ablation study on self-correction with extended context length and revision iterations on the MiniF2F test split at pass@32.

- The results show that removing compiler feedback significantly lowers performance, confirming that specific error messages are crucial for effective revision.
- Removing the reasoning from previous attempts also slightly degrades performance, indicating that retaining the chain-of-thought from prior rounds is beneficial.
- With an extended context and more revision iterations, the full self-correction models pass@32 accuracy on MiniF2F reaches an average of 92.7%, which surpasses the 92.2% performance of the model without self-correction at pass@8192.

RL and Model Averaging

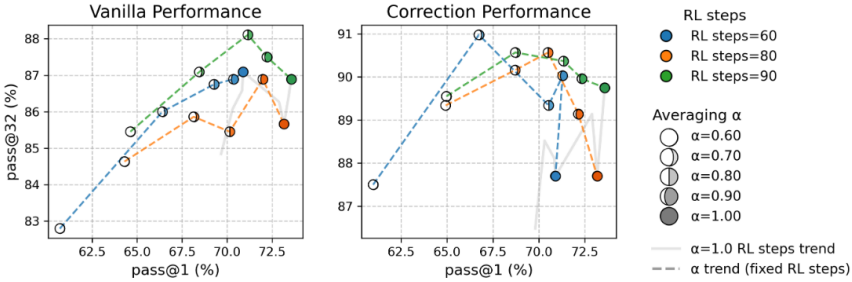


Figure 8: The effects of varying RL steps and model averaging ratios on the pass@1 and pass@32 performance of models, both with and without correction.

- $\theta_{\text{new}} = (1-\alpha)\theta_0 + \alpha\theta$
- Pass@1 consistently increases with α and RL steps. Model averaging mainly improves pass@32.

Outline

Discussion

Main Contributions

- The multi-turn style self-correction is superior to large scale sampling in computing budget, which Seed-Prover also claims.
- Since there have been open-sourced provers that are powerful enough, formal proof data synthesis can alleviate the data scarcity problem.
- Model averaging can improve the diversity of the model.

Discussion

- The tool-assisted RL approach is deprecated in this paper for engineering problems and algorithmic uncertainty. But we can validate it now.
- The model averaging technique is tricky. The reference paper says that the zero-shot model and fine-tuned model are connected by a linear path in the weight-space along which accuracy remains high.