Target-Based Automated Conjecturing for Neural Theorem Proving

Marco Dos Santos [1] **Albert Q. Jiang** [1] **Wenda Li** [2] **Mateja Jamnik** [1]

Abstract

Neural theorem proving models often struggle with hard problems, in part due to the lack of training examples. We introduce a novel approach to automated conjecturing that increases the amount of training examples by generating conjectures specifically tailored to a target set of difficult problems. Our method is an evolutionary algorithm in which a conjecturer iteratively proposes new conjectures, guided by feedback on how much they help a prover make progress on the target set. Conjectures that most effectively enhance the prover's performance become seeds for the next iteration. We demonstrate that this approach significantly enhances the prover's performance on hard problems, with preliminary results showing an increase in proof success rate from 8% to 29% on a target set of complex number problems after only five iterations.

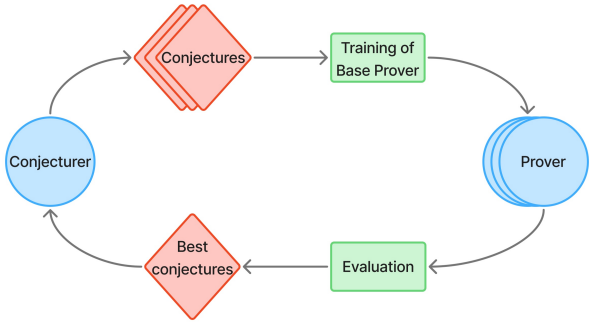

Figure 1. One iteration of our evolutionary algorithm: (1) The Conjecturer generates one set of conjectures per seed theorem. (2) Each set is used to train a separate instance of the base Prover with reinforcement learning. (3) Each Prover instance is evaluated on a fixed target set of theorems; scores are assigned based on performance gains. (4) A maximum coverage algorithm selects the highest-scoring conjectures across all sets. (5) These conjectures become the new seeds, and the base Prover is updated with them.

1. Introduction

The past few years have seen rapid progress in neural theorem proving models, illustrated by the increasing success of the state-of-the-art systems on the miniF2F benchmark (Zheng et al., 2022), which went from 36.6% to 88.9% (Polu et al., 2023; Ren et al., 2025), as well as by AlphaProof's performance at the 2024 IMO (AlphaProof and AlphaGeometry teams, 2024). These advances rely on reinforcement learning over large corpora of formal problems, often constructed via autoformalization (Wang et al., 2018; Szegedy, 2020; Wu et al., 2022), which converts informal problems into formal ones. For instance, AlphaProof (AlphaProof and AlphaGeometry teams, 2024) was trained on 100M autoformalized problems. While autoformalization enables the expansion of formal mathematical data, it remains inherently constrained by the availability and scope of existing human-written mathematics. In advanced mathematics, human-written data is still sparse relative to the complexity and breadth of the field, which poses an important limitation for improving theorem proving capabilities beyond olympiad-style and undergraduate level problems. Automated conjecturing offers a promising path to overcome these limitations by generating, independently of human-written data, new formal statements for training provers.

Unlike prior work resulting in the generation of difficult (Poesia et al., 2024; Dong & Ma, 2025) or human-like conjectures (Wang & Deng, 2020), our method takes an outcome-driven approach by conditioning conjecture selection on a target set of theorems. Our goal is to produce conjectures that increase in complexity and also align more closely with the target theorems, ultimately improving the neural prover's performance on this specific target set. Figure 1 illustrates our approach. Our method enables the prover model to solve problems it could not initially prove, using synthetic conjectures tailored to the specific target distribution. The initial results demonstrate an improvement in the prover's success rate from 8% to 29% in just five iterations.

*Equal contribution [1]University of Cambridge [2]University of Edinburgh. Correspondence to: Marco Dos Santos <mjad3@cam.ac.uk>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

2. Method

Similar to prior work by Poesia et al. (2024); Dong & Ma (2025), our method is based on an evolutionary algorithm. At each iteration, conjectures are selected based on the Prover's performance on a target set.

2.1. Evolutionary Algorithm

The training procedure interleaves conjecture generation, reinforcement learning of the Prover model, and evaluation. The goal is to progressively generate useful conjectures that help the Prover solve a fixed set of hard problems. This process is illustrated in Figure 1.

The evolutionary algorithm begins with four components: a base Prover model, a Conjecturer model, an initial set of seed problems that the Prover can already solve, and a target set of hard problems that the Prover currently cannot solve.

Each iteration of the training loop goes as follows:

1. **Conjecture generation:** The Conjecturer generates one set of conjectures for each seed problem.
2. **Reinforcement learning on conjectures:** A separate Prover instance is trained on each set of generated conjectures, using the GRPO algorithm (Zhihong Shao, 2024).
3. **Evaluation on the target set:** Each of the resulting Prover instances is evaluated on the target set by generating k proof attempts per problem, and recording the problem-wise performance of each instance (pass@k and number of proofs found out of k attempts).
4. **Conjecture selection:** The conjectures associated with the highest-performing Prover instances are selected, prioritising those that improve performance on previously unsolved problems over raw performance. That is, weaker sets of conjectures that allowed the Prover to solve new target problems are preferred over stronger ones that only reinforce already-solved cases. The selection is constrained such that the total number of conjectures that led to successful updates in step 3 does not exceed the number of seeds for the next iteration. These selected conjectures become the new seeds.
5. **Base Prover update:** The selected conjectures are used to further train the base Prover with GRPO. These conjectures also become the seeds for the next iteration. This updated model becomes the new base Prover, and the loop restarts from step 1.

Full details on the training of the Prover and Conjecturer, as well as the selection of seeds and the target set, are provided in Appendix A. Examples of selected conjectures are provided in Appendix D.

3. Experiments

We conducted experiments to assess the effectiveness of our method. Implementation details for the base Prover, the Conjecturer, and the seed and target problems are provided in Appendix B.

Table 1 reports the results after running five iterations of the evolutionary loop.

Iteration	Pass@16 on target set	Total proofs
0 (baseline)	8.2%	5
1	18.4%	19
2	20.5%	22
3	20.5%	41
4	20.5%	36
5	28.6%	46

Table 1. Performance of the Prover on the target set across evolutionary iterations. Iteration 0 corresponds to the base Prover obtained after supervised fine-tuning. The table reports: (1) the number of target problems (out of 49) for which at least one proof was found out of 16 attempts, (2) the corresponding percentage, and (3) the total number of successful proof attempts across all 49 problems, each attempted 16 times. Due to evaluation randomness, we observed that the base Prover was able to solve 4 distinct problems (8.2%) with 5 total successful attempts, which accounts for the non-zero baseline shown in the table.

4. Conclusion and Future Work

We introduced a novel approach to automated conjecturing designed to improve the performance of a Prover model on a specific target set of problems. Our method generates conjectures conditioned on seeds, uses reinforcement learning and evaluation of the Prover to score them, and selects those that most effectively enhance performance on the target set. Preliminary results show that this approach significantly improves the Prover's performance on a target set of complex number problems after only five iterations.

Several directions for future work are promising. The method could be scaled further by increasing the number of conjectures, the number of seeds, or using larger models. Selection of seeds and target problems could be made automatic by embedding problems with the Conjecturer or Prover model and applying clustering techniques, selecting the target set as a narrow distribution to enable learning along a coherent path rather than multiple unrelated trajectories, and selecting the seeds as representatives of diverse clusters to ensure broad coverage of the distribution of easy problems. We also plan to test the generality of our method by applying it to diverse target sets. We envision that our method could be combined with the test-time reinforcement learning approach from AlphaProof and AlphaGeometry teams (2024) by enhancing a single difficult theorem into a larger target set, which could then be used by our method.

References

AlphaProof and AlphaGeometry teams. AI achieves silver-medal standard solving International Mathematical Olympiad problems. Blog post, Google DeepMind, 2024. URL https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/.

Dong, K. and Ma, T. STP: Self-play LLM Theorem Provers with Iterative Conjecturing and Proving. *arXiv e-prints*, pp. arXiv–2502, 2025.

Dos Santos, M., Wang, H., de Saxc, H., Wang, R., Baksys, M., Unsal, M., Liu, J., Liu, Z., and Li, J. Kimina Lean Server: Technical Report, 2025. URL https://arxiv.org/abs/2504.21230.

Johnson, J., Douze, M., and Jégou, H. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7 (3):535–547, 2019.

Poesia, G., Broman, D., Haber, N., and Goodman, N. Learning formal mathematics from intrinsic motivation. *Advances in Neural Information Processing Systems*, 37: 43032–43057, 2024.

Polu, S., Han, J. M., Zheng, K., Baksys, M., Babuschkin, I., and Sutskever, I. Formal Mathematics Statement Curriculum Learning. In *ICLR*, 2023.

Ren, Z., Shao, Z., Song, J., Xin, H., Wang, H., Zhao, W., Zhang, L., Fu, Z., Zhu, Q., Yang, D., et al. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.

Szegedy, C. (ed.). *A Promising Path Towards Autoformalization and General Artificial Intelligence*, 2020.

Wang, H., Unsal, M., Lin, X., Baksys, M., Liu, J., Santos, M. D., Sung, F., Vinyes, M., Ying, Z., Zhu, Z., Lu, J., de Saxc, H., Bailey, B., Song, C., Xiao, C., Zhang, D., Zhang, E., Pu, F., Zhu, H., Liu, J., Bayer, J., Michel, J., Yu, L., Dreyfus-Schmidt, L., Tunstall, L., Pagani, L., Machado, M., Bourigault, P., Wang, R., Polu, S., Barroyer, T., Li, W.-D., Niu, Y., Fleureau, Y., Hu, Y., Yu, Z., Wang, Z., Yang, Z., Liu, Z., and Li, J. Kimina-Prover Preview: Towards Large Formal Reasoning Models with Reinforcement Learning, 2025. URL https://arxiv.org/abs/2504.11354.

Wang, M. and Deng, J. Learning to prove theorems by learning to generate theorems. *Advances in neural information processing systems*, 33:18146–18157, 2020.

Wang, Q., Kaliszyk, C., and Urban, J. First Experiments with Neural Translation of Informal to Formal Mathematics. In Rabe, F., Farmer, W. M., Passmore, G. O., and Youssef, A. (eds.), *Intelligent Computer Mathematics*, pp. 255–270, Cham, 2018. Springer International Publishing. ISBN 978-3-319-96812-4.

Wu, Y., Jiang, A. Q., Li, W., Rabe, M., Staats, C., Jamnik, M., and Szegedy, C. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368, 2022.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Zheng, K., Han, J. M., and Polu, S. miniF2F: a cross-system benchmark for formal Olympiad-level mathematics. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9ZPegFuFTFv.

Zhihong Shao, Peiyi Wang, Q. Z. R. X. J. S. M. Z. Y. L. Y. W. D. G. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, 2024. URL https://arxiv.org/abs/2402.03300.

A. Method Details

A.1. Supervised Training of the Prover Model

The base Prover is initialised by training a large language model with supervised learning on a large dataset of formal proofs. Each training example consists of a formal statement paired with its corresponding formal proof. Following Wang et al. (2025); Ren et al. (2025), the model is trained to generate the full proof from the statement, rather than predicting the next tactic step-by-step.

A.2. Supervised Training of the Conjecturer Model

A central contribution of our method is a novel supervised training strategy for the Conjecturer model, by simplifying and generalising the approach introduced by Dong & Ma (2025). The Conjecturer is trained to generate new conjectures conditioned only on a formal statement and its proof, without requiring additional prompts or auxiliary lemmas.

The objective is to produce conjectures whose proofs are close, in topic and difficulty, to the proof of the input. This ensures that the evolutionary process proceeds through smooth transitions: if the Prover can solve the input, it should also be able to solve the generated conjecture. For this to hold, similarity must be defined from the Prover's perspective. The Prover model's internal embeddings of proofs are used to identify similar problems in its training data, and a dataset of such pairs is constructed for training the Conjecturer. This allows the Conjecturer to learn how to generate conjectures that lie close to the input, both in terms of difficulty and topic, from the Prover's perspective.

To construct the training dataset, formal proofs are embedded using the last hidden states of the Prover model. For each (statement, proof) pair, a set of neighbours above a given cosine similarity threshold is retrieved. The number of neighbours is capped to ensure diversity, resulting in training pairs of the form (statement1+proof1, statement2), where the proof of statement2 is similar to proof1.

A.3. Selection of Seeds and Target Set

An important aspect of our method is the choice of the target set and the initial seed problems. For the optimisation signal to be strong, the target set must be large enough to provide a stable learning signal, and narrow enough in distribution so that the system can progressively improve along a coherent path. In contrast, a wide target distribution would require learning multiple unrelated trajectories, making training less efficient. The seed problems, which must already be solvable by the Prover, should cover a broad range of topics to serve as good representatives of the easy problem distributions.

The selection of seed and target problems depends on the Prover's success rate across different problems. The target set must contain difficult problems from a narrow distribution to enable the system to improve along a coherent path, rather than learning multiple unrelated trajectories, while still being large enough to provide a stable optimisation signal. It is constructed from problems that are consistently unsolved by the Prover. On the other hand, the seed set consists of problems that the Prover can already solve reliably and that cover a broad range of topics, to serve as good representatives of the easy problem distribution.

To make this process more principled and scalable, future work could use clustering techniques over problem embeddings derived from the Prover or Conjecturer. The seed set could consist of representatives from diverse clusters, and the target set of a tight cluster around a specific area of interest. This strategy could also be used to tackle a single hard problem by generating multiple variations to form a synthetic target set, complementing techniques such as test-time reinforcement learning, explored by AlphaProof and AlphaGeometry teams (2024).

B. Implementation Details

B.1. Supervised Training of the Prover

We initialise the base Prover by fine-tuning Qwen2.5-Math-1.5B (Yang et al., 2024) on a dataset of 130,000 formalized proofs of competition-style problems, provided by Numina. Each example consists of a formal statement paired with a formal proof. We train the model for one epoch, using a batch size of 8, a learning rate of 5e-5, and a cosine learning rate scheduler.

B.2. Supervised Training of the Conjecturer

To build the training dataset, we embed the 130,000 proofs from the Prover's training dataset using the last hidden states of the Prover model. We then use the Faiss library (Johnson et al., 2019) to efficiently retrieve similar embeddings: for each (statement, proof) pair, we select all other pairs with at least 0.97 cosine similarity. This threshold ensures that all the pairs are meaningfully similar, while also guaranteeing broad coverage: 93% of the original samples had at least one neighbour above 0.97 cosine similarity, meaning that most samples appear in the resulting dataset at least once. We cap the number of retrieved neighbours to five per sample, resulting in a total of 120,000 (statement1+proof1, statement2) pairs, where the proof of statement2 is similar to proof1.

To train the Conjecturer, we fine-tune Qwen2.5-Math-1.5B (Yang et al., 2024) on this dataset for one epoch, with a batch size of 8, a learning rate of 5e-5 and a cosine learning rate scheduler.

B.3. Selection of Seeds and Target Set

To select the initial seeds and the target set, we first evaluate the base Prover on a held-out test set of 2,000 problems, generating 16 proof attempts per problem. In our experiments, we use the Kimina Lean Server (Dos Santos et al., 2025). Among these 2,000 problems, 54% are completely unsolved by the Prover, while 46% are solved at least once. Only 15% of the problems are solved reliably at least 8 times out of 16 attempts.

As explained in Section A.3, the target set must contain difficult problems from a narrow distribution to enable the system to improve along a coherent path, rather than learning multiple unrelated trajectories, while still being large enough to provide a stable optimisation signal. After manual inspection of the 2,000 problems, we identified unsolved problems about complex numbers as satisfying both criteria: they form a sufficiently narrow domain, and there are enough of them to ensure a smooth signal. The target set is therefore constructed by selecting all complex number problems for which the Prover failed all 16 attempts, resulting in a total of 49 problems.

Since the seed problems must already be solvable by the Prover and cover a broad range of topics to serve as good representatives of the easy problem distribution, we randomly select 13 problems that the Prover solves reliably, at least 8 times out of 16. To check whether the system preferentially selects conjectures from domain-relevant seeds, we also include three complex number problems meeting the same criterion. This gives us an initial pool of 16 seeds to bootstrap the evolutionary loop.

Examples of selected seed and target problems are provided in Appendices C.1 and C.2.

C. Examples of Seed and Target Problems

C.1. Examples of Seed Problems

Example 1
Lean (formal statement and proof):

```
theorem algebra_488298 (f : ℝ → ℝ) (hf : f = fun x => 3 ^ x + x - 3) :
    (f 0) * (f 1) < 0 := by
  rw [hf]
  norm_num
```

Corresponding natural language statement:

Let $f : \mathbb{R} \to \mathbb{R}$ defined by $f(x) = 3^x + x - 3$.

Show that:

$$f(0)f(1) < 0$$

Example 2
Lean (formal statement and proof):

```
theorem algebra_339741 (x : ℝ) (h : Real.sin x = -2 / 3) :
    Real.cos (2 * x) = 1 / 9 := by
```

```
  rw [Real.cos_two_mul]
  rw [← sub_eq_zero]
  nlinarith [Real.sin_sq_add_cos_sq x, sq_nonneg (Real.cos x)]
```

Corresponding natural language statement:

Let $x \in \mathbb{R}$ such that $\sin(x) = -\frac{2}{3}$.

Show that:

$$\cos(2x) = \frac{1}{9}$$

Example 3
Lean (formal statement and proof):

```
theorem algebra_149850 (x y : ℕ) (h₀ : x > 0) (h₁ : y > 0)
    (h₂ : x + 10 = y) (h₃ : y + 10 = 2 * x) :
    x = 20 := by
  omega
```

Corresponding natural language statement:

Let $x, y \in \mathbb{N}$, with $x > 0$ and $y > 0$.

Suppose that $x + 10 = y$ and $y + 10 = 2x$.

Show that:

$$x = 20$$

Example 4
Lean (formal statement and proof):

```
theorem algebra_426547 {z₁ z₂ : ℂ} (hz₁ : z₁ = 1 + 2 * .I) (hz₂ : z₂ = 1 - .I) :
    (z₁ * z₂).re = 3 := by
  rw [hz₁, hz₂]
  ring_nf
  norm_num
```

Corresponding natural language statement:

Let $z_1, z_2 \in \mathbb{C}$ defined by $z_1 = 1 + 2i$ and $z_2 = 1 - i$.

Show that $\Re(z_1 z_2) = 3$

C.2. Examples of Target Problems

Example 1
Lean (formal statement):

```
theorem algebra_388276 (a : ℝ) (z : ℂ) (hz : z = 1 + a * .I) (h : Complex.abs z < 2) :
    -Real.sqrt 3 < a ∧ a < Real.sqrt 3 := by
```

Corresponding natural language statement: Let $a \in \mathbb{R}$ and $z \in \mathbb{C}$ defined by $z = 1 + ai$.

Suppose that $|z| < 2$.

Show that:

$$-\sqrt{3} < a < \sqrt{3}$$

Example 2
Lean (formal statement):

```
theorem algebra_341547 {z : ℂ} (hz : z = (Complex.I ^ 2016) / (3 + 2 * Complex.I)) :
    z.re > 0 ∧ z.im < 0 :=
```

Corresponding natural language statement: Let $z \in \mathbb{C}$ defined by $z = \frac{i^{2016}}{3+2i}$.

Show that the real part of z is positive and the imaginary part is negative.

Example 3
Lean (formal statement):

```
theorem algebra_424944 (a : ℝ) (ha : a ≠ 0) :
    (∃ b : ℝ, (a + Complex.I) / (1 - Complex.I) = b * Complex.I) ↔ a = 1 :=
```

Corresponding natural language statement: Let $a \in \mathbb{R}$, $a \neq 0$.

Show that:

$$\left(\exists b \in \mathbb{R}, \frac{a+i}{1-i} = bi\right) \iff a = 1$$

D. Examples of Selected Conjectures

Example 1: Selected Conjecture from Iteration 1
Lean (formal statement):

```
theorem algebra_478093 {a b : ℝ} (ha : a + Complex.I * (-b) = 1 / (1 - Complex.I)) :
    a ^ 2 + b ^ 2 = 1 / 2 :=
```

Corresponding natural language statement:

Let $a, b \in \mathbb{R}$ such that $a - ib = \frac{1}{1-i}$.

Show that:

$$a^2 + b^2 = \frac{1}{2}$$

Example 2: Selected Conjecture from Iteration 3
Lean (formal statement):

```
theorem algebra_429360 {x y : ℝ} (hx : x ≠ 0) (hy : y ≠ 0) :
    Complex.I^2 * x - Real.sqrt 2 * Complex.I + y = 0 → x / y = 1
```

Corresponding natural language statement:

Let $x, y \in \mathbb{R}$ with $x \neq 0$ and $y \neq 0$.

Show that if $i^2 x - \sqrt{2}i + y = 0$, then:

$$\frac{x}{y} = 1$$

Example 3: Selected Conjecture from Iteration 5
Lean (formal statement):

```
theorem algebra_491667 {z : ℂ} (hz : z = Complex.I / (-2 + Complex.I)) :
    Complex.abs z = Real.sqrt 5 / 5 :=
```

Corresponding natural language statement:

Let $z \in \mathbb{C}$ such that $z = \frac{i}{-2+i}$.

Show that:

$$|z| = \frac{\sqrt{5}}{5}$$