LEARNING TO DISCOVER INTERESTING MATHEMATICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently, Large Language Models (LLMs) have been increasingly able to solve advanced mathematical problems, including many that have been open for decades. This opens the door to expansion of mathematical knowledge at unprecedented scale. Yet, while LLMs may be able to conjecture and prove more and more theorems, it remains open whether this new mathematical knowledge is *interesting* or *useful*. We define intrinsic interestingness of a theorem as the ratio between the length of its proof and the length of its statement. We show that this correlates strongly with an extrinsic measure of the downstream utility of a theorem. We identify the difficulty of a proof conditioned on a set of premises as a useful primitive for computing these metrics, and train a 27B model that predicts proof difficulty more accurately than frontier general-purpose models. Optimizing for our metric creates a model capable of producing more interesting theorems, while also reducing substantial or full overlap with MATHLIB from 91.9% to 30.6%, showcasing the creation of more out-of-distribution math. We show that our system can generate candidate theorems, select the most interesting among them, and iteratively build on a self-expanding mathematical library. These metrics provide a practical and quantifiable signal for ranking conjectures and guiding proof search within formal mathematical libraries. Our framework provides a path towards self-expanding, machine-verified mathematical libraries that can choose worthwhile statements without relying on human-supplied targets.

1 INTRODUCTION

Since the earliest work on the subject, mathematical reasoning has been a cornerstone of research in Artificial Intelligence (AI) (Turing, 1939; Newell & Simon, 1956). Recent work on applications of Large Language Models (LLMs) to solving mathematical conjectures has shown striking success, and we now have seen many instances of LLMs solving problems that have eluded human mathematicians (Novikov et al., 2025; Sothanaphan, 2026; Alexeev et al., 2026a;b; Alon et al., 2026; Chen & Jiang, 2026; Oum, 2026; Huang, 2026; OpenAI, 2026b). Though LLM-based theorem provers can often make errors, they are particularly strong and reliable when their outputs are verified by a proof assistant such as Lean 4 (de Moura & Ullrich, 2021; The mathlib Community, 2020; Yang et al., 2023; Xin et al., 2024; AlphaProof and AlphaGeometry teams, 2024; Ren et al., 2025). However, a common critique of applications of LLMs to solving mathematical conjectures is that they often only appear to solve problems that lie within the "convex hull" of existing mathematics.

Our long-term goal is to build systems capable of *autonomous mathematical discovery*. Such a system should be able to engage in *open-ended* reasoning. Starting from a set of initial premises, the system should be able to formulate new questions, decide which are worth pursuing, construct and verify their proofs, and build on this knowledge (Barkeshli et al., 2026). A fundamental challenge to achieving such a system is the question of interestingness of the mathematical statement. The version of mathematics that was created by human intuition over thousands of years has proven to be an "unreasonably effective" tool to approaching real world problems (Wigner, 1960). In contrast, arbitrarily adding trivial statements, or even nontrivial but *uninteresting* statements, is unlikely to lead to the same success that mathematics has achieved in the past.

> *"Science is built up with facts, as a house is with stones. But a collection of facts is no more a science than a heap of stones is a house." – Science and Hypothesis,* Poincaré (1905)

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We can reason by analogy with Borges's "Library of Babel" (Borges, 1962; Litt, 2026). Borges's library holds every book that can be written, and therefore holds every truth in it; it is nonetheless useless, since no reader could find meaning within its nearly infinite shelves. The space of all true mathematical statements has the same character, whereas mathematics, as it has been created by humans, does not. Extrapolating outside the "convex hull" of mathematics, therefore, requires both the ability to navigate the space of possible deductions and a notion of value that distinguishes a discovery from a merely valid statement.

We seek a system that can begin with the mathematics that is already known, and recursively produce new mathematical results. Any such autonomous system needs a quantitative object to optimize. Prior work has often focused on "Intrinsic Motivation" (Poesia et al., 2024; Tsoukalas et al., 2025). We take a different view, one in which there are inherent aspects of mathematics that can quantify the quality of a mathematical statement, its *interestingness*. Prior work has found that LLMs do not robustly capture the same notions of interestingness, and the same diversity in those notions, as humans do (Mishra et al., 2025). In the present work, we consider two complementary sources of value. On one hand, a theorem may be *intrinsically interesting*, if it is easy to state but very difficult to prove. Alternatively, a theorem could be *useful* because of its relation to the surrounding body of mathematics. A useful theorem is a reusable abstraction that - once added to a body of mathematics - compresses later proofs by making them easier to derive.

A fundamental primitive required to understand or estimate either of the above quantities is the difficulty to prove a theorem T, conditioned on a set of premises P which are already available, which we will term $V(T|P)$. Proof assistants (de Moura & Ullrich, 2021) allow a way to quantify the difficulty of a proof, in terms of the number of lines of code required to formally prove it. Lean 4's library MATHLIB provides a large dependency graph of machine-checked definitions, theorems, and proofs (The mathlib Community, 2020), which we can utilize to learn to model the conditional proof difficulty $V(T \mid P)$, mathematical interestingness and utility, and autonomous mathematical discovery. We make the following primary contributions.

Contributions.

1. We formalize proof difficulty as the computational cost of deriving a target theorem from a given mathematical context in Lean 4. We post-train a 27B parameter LLM to predict the difficulty of a proof given a set of premises, and show that this predictor outperforms frontier LLMs.
2. We motivate and define a notion of a theorem's *interestingness* as the ratio between the length of the proof and the length of the statement of the theorem. We also define a notion of *utility* based on the amount a theorem is able to compress a set of theorems and their proofs, and show that utility strongly correlates with interestingness.
3. With our notion of *interestingness*, we are able to show that we can post-train a language model to produce more interesting theorems, and we find that this quadruples the mean interestingness, as measured by the length of the real proof divided by the length of the statement. This training procedure also reduces the fraction of generated statements judged substantially or fully contained in MATHLIB from 91.9% to 30.6%, showcasing the creation of more out-of-distribution math.
4. We showcase our method in a specific setting where we apply our interestingness metric as an inference-time pruning algorithm to perform recursive mathematical discovery, and find nontrivial statements that are not present in MATHLIB.

2 Modeling the Conditional Proof Difficulty

We ultimately want to train a model that can predict the difficulty of a proof without needing to write the proof beforehand. We model this difficulty as a function of both the target theorem and the premises available in the local context. We define the estimator of the difficulty of proving a theorem T given premises P as the value $V(T \mid P)$. Such a premise-conditioned value function should satisfy certain properties. For any codebase $\mathcal{D}$ of theorems and proofs stated in a formal language like Lean 4, we will denote by $C_{\mathcal{D}}(T|P)$ the ground truth number of lines of code in the codebase to prove theorem T given premises P. For any such codebase, this quantity is not necessarily defined for all choices of T, P, as some theorems may have multiple proofs, so we will define V as satisfying the following conditions:

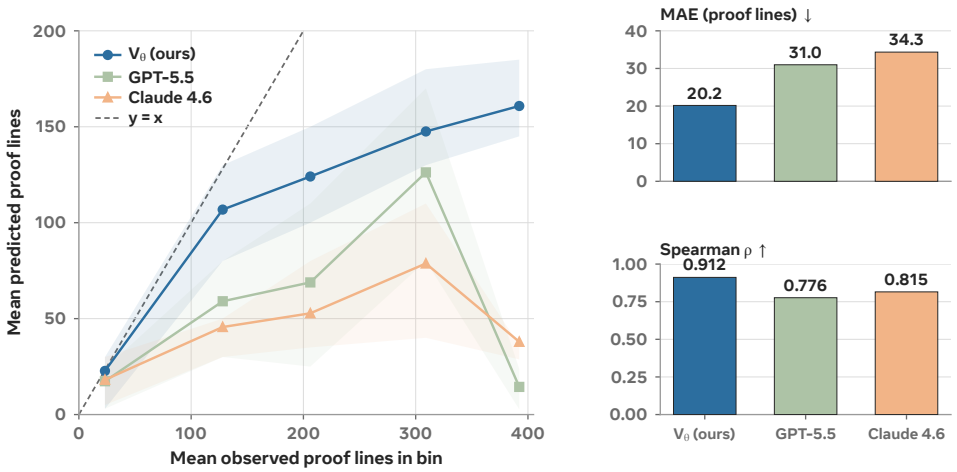


Figure 1: The trained value is substantially better calibrated on held-out samples from MATHLIB. The left panel groups examples by observed proof length, points give the mean observed and mean predicted proof lines, and shaded bands the interquartile range of predictions within each bin. The dashed line represents what we would expect from an optimal predictor. Note that all models underestimate proof length with increased length. The right plots report matched MAE and Spearman ρ on all 4,615 validation prompts.

Definition 1 (Conditional-value axioms)**.** *A conditional proof computational-cost V is* faithful *on $\mathcal{D}$ if for all targets $T \in \mathcal{D}$, lemmas $L \in \mathcal{D}$, and premise sets $P, Q \subset \mathcal{D}$, such that $P \subseteq Q$, the property (A1) below holds whenever $C_{\mathcal{D}}$ is well defined and (A2)–(A3) always hold:*

$$\text{(A1) grounding:} \quad V(T \mid P) = C_{\mathcal{D}}(T \mid P), \tag{1}$$

$$\text{(A2) premise monotonicity:} \quad V(T \mid Q) \leq V(T \mid P), \tag{2}$$

$$\text{(A3) composition:} \quad V(T \mid P) \leq V(L \mid P) + V(T \mid P \cup \{L\}). \tag{3}$$

Axioms (A2)–(A3) hold for the ideal shortest proof. We would expect to see equality in (A3) if a lemma L is required as an intermediate step towards proving the theorem from the premises, giving us the following Bellman-type relation:

$$V(T \mid P) = V(L \mid P) + V(T \mid P \cup \{L\}). \tag{4}$$

We will use these properties to inform the training objective in the following sections.

2.1 Generating a Dataset

We use MATHLIB (de Moura & Ullrich, 2021; The mathlib Community, 2020) as a ground truth library from which we generate our training data. For each theorem we extract its rendered type, the local definitions required to state it, the proof source, and the premises required for the proof. If we were to naïvely train on the dependency graph of MATHLIB we would not get a useful predictor, as the library is built in a very atomized way. But we still need to assign a label to each theorem T. For instance, the median proof length in MATHLIB is 3 lines of code, and the median number of times any particular premise is used is 1. To remedy this, we employ a technique we call **premise expansion**. We call a statement L a premise of T if it is *directly* used in the proof of T, and we will write P or $P(T)$ to denote the set of premises of T. In this case, one *expansion step* will remove L from the set of premises of T, and add , $P(L)$, the premises of L to the premises of T. So the new set of premises of T is $P(T) \cup P(L) \backslash L$. We then add the number of lines of code in the proof of L to the label of T. Iterating moves the visible frontier backward through the dependency Directed Acyclic Graph (DAG)(Appendix B.1), creating a richer dataset of theorem, premises, proof triplets. We end up with around 100k data points which we use in the next section. We evaluate on a withheld evaluation set coming from the same construction.

2.2 Modeling the Conditional Proof Length

We want to train a model V_θ that can predict the difficulty of proving a theorem T conditioned on a set of premises P. We fine-tune Qwen3.6-27B (Qwen Team, 2026) with group relative policy optimization (GRPO) (Shao et al., 2024; Miles Team, 2026) on the dataset from Section 2.1 for 350 steps. Full definitions and specific details on the training setup are relegated to Appendix B.2. We create a reward that is designed to satisfy the desiderata in Definition 1. We design $\mathcal{L}_{\text{truth}}$ to satisfy (1). If L is a lemma used to go from P to T, in order to satisfy (4), we introduce $\mathcal{L}_{\text{Bellman}}$. To satisfy (2), we add $\mathcal{L}_{\text{drop}}$, where $Q \subsetneq P$. Our final reward is,

$$\begin{aligned}\mathcal{R} = &- |\log V_\theta(T|P) - \log C_{\mathcal{D}}(T|P)| && (\mathcal{L}_{\text{truth}})\\ &- |\log V_\theta(T|P) - \log(V_\theta(L|P) + V_\theta(T|P \cup \{L\}))| && (\mathcal{L}_{\text{Bellman}})\\ &- \max\{0, \log V(T|P) - \log V(T|P\backslash Q)\} && (\mathcal{L}_{\text{drop}})\end{aligned}$$

We evaluate models on a held-out validation set of 4,615 labeled validation prompts. All models receive identical definitions, premises, target, and output instruction. We report mean absolute error (MAE) and Spearman rank correlation, computing both only over parsed non-negative answers (Appendix B.3). Figure 1 showcases performance of our trained model over GPT-5.5 (OpenAI, 2026a) and Claude Opus 4.6 (Anthropic, 2026). We find that while all three models tend to underestimate the difficulty of proofs, ours is substantially more accurate and better calibrated than the others.

3 Interestingness-Driven Mathematical Discovery

A human mathematician may find a mathematical conjecture valuable for many reasons. They could be interested in a statement because of historical or social factors, like for instance that several famous mathematicians before them tried and failed to find a proof of a statement. They could also study statements that are related to the physical world. These are factors which are extrinsic to the field of math, and rely on its relation to the world outside it. As such, these factors are not something that we could model in search of a quantitative definition of the interestingness of a statement; they lie outside the scope of our present objective.

We instead focus on structural properties that can be measured directly in a formal mathematical library. In particular, statements that are simple to express but difficult to prove often align with human intuitions about mathematical interestingness. Motivated by this observation, we define an intrinsic notion of interestingness that combines statement length with conditional proof difficulty. This quantity is not intended to capture every source of mathematical value, but to provide a concrete objective for guiding mathematical discovery. We separately consider a theorem's utility, which measures its value to subsequent mathematical developments.

In Section 3.1 we define and study the *interestingness* of a mathematical statement. In Section 3.2 we examine the *utility* of a mathematical statement and show its relation to the interestingness. We then show in Section 3.3 that interestingness is a concrete metric that can be optimized, via training and at inference-time.

3.1 The Intrinsic Interestingness of a Mathematical Statement

In this section, we attempt to define a notion of the interestingness of a mathematical statement or conjecture that aims to capture an *intrinsic* property of the mathematical result, ignoring any relation to the outside world, or to other parts of the mathematical literature. We claim that a statement T is *interesting* given a set of premises P if it is easy to state, and has a long and incompressible proof. To compute this, we can take a ratio between the length of a proof given some premises in terms of lines of code, and the number of characters of Lean 4 needed to state a theorem given some set of definitions one already has access to. For instance, one would expect that a theorem in probability theory is easy to state for a reader who already has a measure-theoretic vocabulary and enormously difficult for a reader given access to only statements in algebraic geometry. We recall the following quote by Pólya on the nature of aesthetics in math,

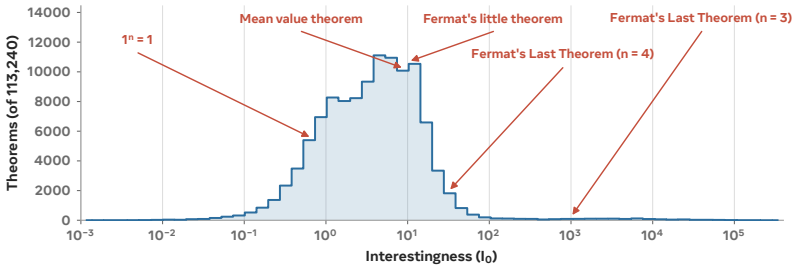


Figure 2: Quantifying the interestingness of a statement. Here we show the distribution of interestingness I_0, as defined in Section 3.1, of all statements from MATHLIB. The resulting ordering is intuitive, with basic algebraic identities at the bottom, analysis in the middle because its definitional prerequisites are enormous, and theorems that are famously easy to state but hard to prove, like Fermat's Last Theorem for exponent 3, at the top.

> *"The elegance of a mathematical theorem is directly proportional to the number of independent ideas one can see in the theorem and inversely proportional to the effort it takes to see them." – Mathematical Discovery,* Pólya (1981)

Formally, for any Lean declaration X, let $S(X)$ denote the number of characters in its statement, and let $C(X)$ be the set of all definitions required to state X, including whatever definitions are needed to state those definitions, recursively. So $C(X)$ will contain all the definitions required to formally state X from the axioms. We will then define the conditional description length as

$$L(T \mid P) \;=\; S(T) + \sum_{d \in C(T) \setminus C(P)} S(d). \tag{5}$$

In this way, the term $L(T|P)$ counts the description length of all the definitions needed to state the theorem T, given access to all the definitions used in the premises P. We then formally define the conditional interestingness[1] of a statement as

$$I(T \mid P) \;=\; 100\, \frac{V(T \mid P)}{L(T \mid P)}. \tag{6}$$

We choose this definition to quantify the intrinsic difficulty of a mathematical statement as it is trivially possible to construct arbitrarily difficult statements, if we do not normalize by the description length of the statement. For instance, suppose we had a fixed set of premises P and a set of theorems $\{T_1, T_2, \ldots\}$ that are all entirely unrelated and have a fixed proof length. Then the "theorem" $\hat{T}_n = T_1 \wedge T_2 \wedge \cdots \wedge T_n$ would have difficulty $V(\hat{T}_n|P) = O(n)$, whereas the interestingness is $I(\hat{T}_n|P) = O(1)$.

In the special case where we consider the empty set of premises, we can create an absolute scale of the interestingness of a mathematical statement $I_0(T) = I(T|\varnothing)$[2]. With no premises, the denominator is the target's own vocabulary length, $L(T \mid \varnothing) = S(T) + \sum_{d \in C(T)} S(d)$. As we no longer require conditioning on a set of premises, the numerator can now be deterministically computed from a library such as MATHLIB, as we can just unroll the proof as written via premise expansion. The result (Figure 2) serves as a useful verification for the construction of our metric on existing statements of MATHLIB. We can see in the bottom decile simple algebraic relations, like for instance the identity that $1^n = 1$, and in the upper quartiles we see results such as special cases of Fermat's Last Theorem, which happen to require very little machinery to state but are tough to prove.

3.2 Intrinsic Interestingness Correlates with Extrinsic Utility

We will define the *utility* of a theorem to answer the following question: If we had access to this statement for free, how much would it compress the rest of the mathematical library? Concretely, we

[1] We use a factor of 100 in the definition here so that the Interestingness has a mean $\tilde{1}$, as in Figure 2 and Figure 4.

[2] This corresponds to the notion of "interest" from Aksenov et al. (2026).

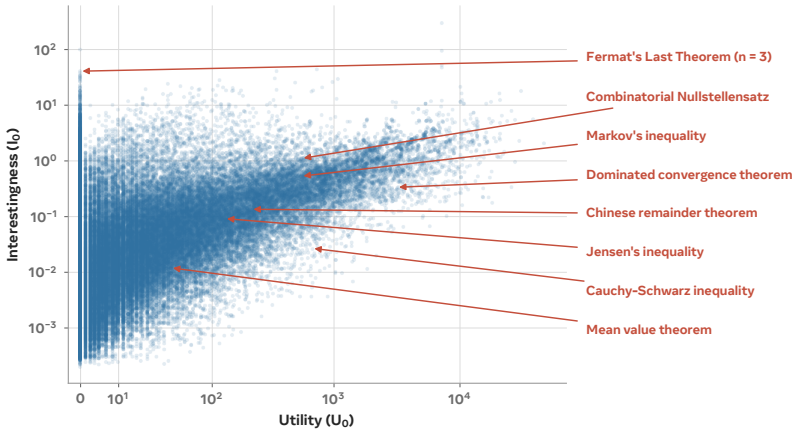


Figure 3: Utility and interestingness are correlated. Each point is a MATHLIB theorem. Utility U_0, defined in Section 3.2, is the number of lines of code saved across the library when the theorem is admitted as a premise, while interestingness I_0, defined in Section 3.1, is the ratio between a theorem's proof length and description length. Excluding the declarations with $U_0 = 0$ gives a Spearman $\rho = 0.756$.

compare two libraries: one in which T is available as a premise, and one in which it is not, so that every statement relying on T must re-derive it from scratch. The utility of T is the number of lines of code saved by the first library relative to the second. If we let $D(T)$ be the set of theorems and lemmas that cite / use T directly, then $|D(T)|$ would be the number of direct users of a theorem. So we can define

$$U_0(T) = |D(T)| \cdot V(T|\varnothing). \tag{7}$$

Despite U_0 being an *extrinsic* measure, depending on a theorem's relation to the broader mathematical theory surrounding it, and I_0 being an *intrinsic* measure depending only on a theorem and its proof, we find that they are strongly correlated. In Figure 3, we see a strong positive correlation between U_0 and I_0 on MATHLIB, giving a Spearman correlation of 0.756 if we disregard statements with no downstream users. While some statements have high interestingness but low utility, few statements, if any, have high graph utility without high interestingness. Interestingness is therefore a practical surrogate for the quality of a mathematical statement as it can be computed or estimated at proposal time, whereas utility is observable only after placing a theorem in a body of work and seeing its descendants. We recall the following quote of Thurston.

> *"Our aesthetic instincts draw us to mathematics of a certain depth and connectivity. The very depth and beauty of the patterns makes them likely to be manifested, in unexpected ways, in other parts of mathematics, science, and the world." – Mathematical Education,* Thurston (2005)

3.3 Optimizing for Interestingness

The preceding construction turns the task of creating new, interesting, mathematics into an optimizable objective. Given a mathematical context of prior lemmas and premises P, a conjecturing model, or *conjecturer*, should be able to propose a statement T which is cheap to express in the vocabulary of P but requires a difficult proof. With the use of our estimator V_θ we trained in Section 2, we can obtain an estimate of the conditional interestingness, which we can use as a reward signal to train a model to produce more interesting conjectures. From the MATHLIB training split we randomly select $10{,}000$ sets of premises. Each set P contains at least 16 premises, with the median containing 77. Appendix C.1 details the sampling procedure. The model is conditioned on the available premises and definitions and emits a standalone Lean proposition. For a valid and nontrivial proposal, the training reward is,

$$R(T, P) = 0.25 + \log\left(1 + \frac{V_\theta(T \mid P)}{L(T \mid P)}\right) = 0.25 + \log\left(1 + \frac{I_\theta(T \mid P)}{100}\right). \tag{8}$$

Here V_θ is a frozen proof line count predictor and L is the conditional description length from Equation 5. The logarithm controls the influence of outliers while preserving the ordering induced

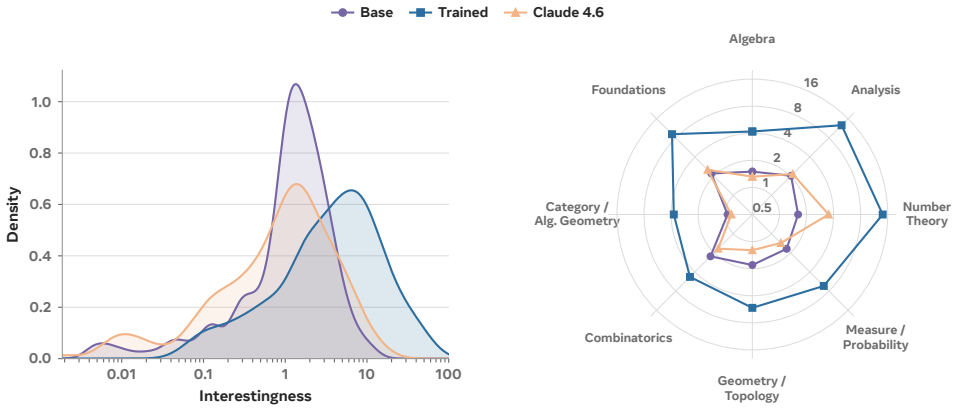


Figure 4: Training shifts generated theorems toward higher ground-truth interestingness. Our trained 27B model is consistently able to produce more interesting statements, with 20 statements per coarse area. Left: kernel densities estimator (KDE; a method that smooths the observed histogram with a kernel to give a continuous density estimate) fitted in $\log_{10}$-interestingness space over all 160 outcomes per model. Right: mean ground-truth interestingness by mathematical area over the same complete cohort, shown on a logarithmic radial scale from 0.5 to 20. Colors identify the same models in both panels.

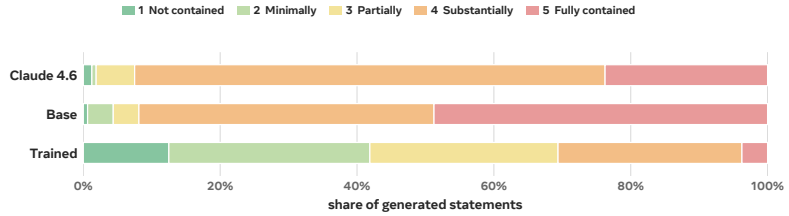


Figure 5: Interestingness training reduces overlap with existing MATHLIB content. For each model we take the 160 statements from Figure 4, and judge to what degree each statement is contained within MATHLIB. Appendix C.2.2 gives the rubric, prompt, and protocol.

by interestingness. Parse failures receive reward -0.5, Lean-invalid or statements that don't relate to the premises at all receive -0.25, and statements which are able to be proven with simple proof automation tactics[3] receive zero. The constant term 0.25 appears in the reward to provide a baseline reward for valid, relevant, and non-trivial statements. Appendix C.1 describes elaboration, relevance, and triviality checks in full.

We evaluate Qwen3.6-27B (Qwen Team, 2026), trained with 75 GRPO steps with the reward from (8), against the corresponding base model and Claude 4.6 on a held-out validation set from eight coarse mathematical areas in MATHLIB. For each conjecturer and area, we repeatedly sample premises and generate candidate theorem statements, then ask a Claude Code with Opus 4.6 proving agent for an exact proof or a tightly constrained marginal repair. Having a proof of each of these statements allows us to verify that they are correct, and to compute the ground-truth proof length, allowing us to get a ground-truth estimate of the interestingness. We retain 20 proven statements per model and area, across 8 areas of mathematics, resulting in 160 total statements per model. We showcase the ground-truth interestingness scores in Figure 4; full details are available in Appendix C.2.

We find that training produces a large shift in ground-truth verified interestingness (Figure 4). Pooled across areas, mean $I(\cdot \mid P)$ rises from 1.76 for the Qwen 3.6 27B base model, to 7.58 for our trained model. The trained model's mean is higher than the base model's in all eight areas, with area-level ratios ranging from $2.10\times$ in combinatorics to $8.72\times$ in number theory. The trained model similarly beats Claude Opus 4.6 (prompted to generate interesting theorems) in all areas.

[3]For instance, `assumption`, `rfl`, `simp`, `tauto`, and `simp_all`.

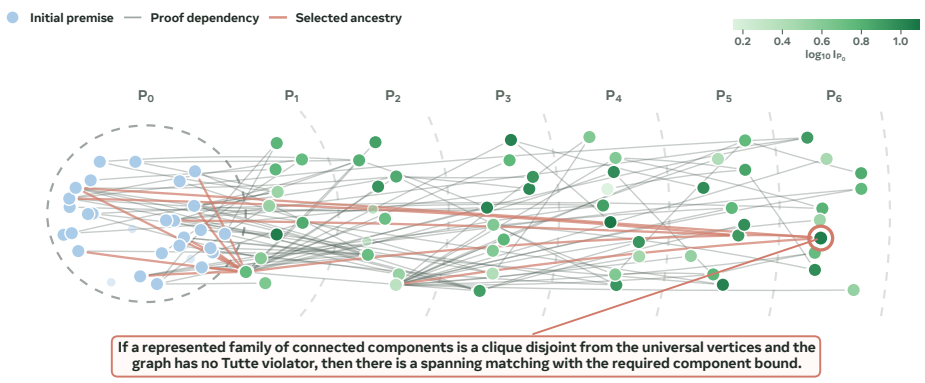


Figure 6: Iterative forward discovery builds a reusable theorem graph. Here we showcase a result from our iterative forward discovery algorithm introduced in Section 3.4. Green shade encodes $\log_{10}$ total interestingness relative to P_0 on a per-figure scale. The red lines trace the ancestry of the most interesting theorem introduced in P_6. More such graphs can be found in Figure 9 in Appendix C.2.

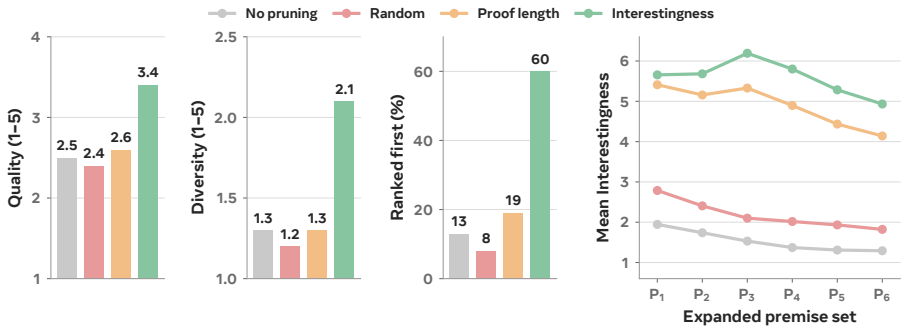


Figure 7: Interestingness pruning is the most effective inference-time pruning criterion. Left three panels: LLM judge scores over all 6 rounds for cohort quality and diversity (Table 4) and the share of four-way comparisons in which each rule's statement is ranked most interesting (Table 3). Right: running mean of the verified interestingness of each set P_n through P_6. More experiments are available in Appendix D.

The shift is also not explained merely by reproducing existing MATHLIB declarations (Figure 5). With the conjecturer and subject area hidden, an LLM-as-a-Judge (in this case Claude Opus 4.6) rates each verified statement on the five-level containment rubric described in the caption. Only 30.6% of statements from the trained model are substantially or fully contained (scores 4–5), compared with 91.9% for Base and 92.5% for Claude. Together, Figures 4 and 5 show that optimizing conditional interestingness moves the conjecturer toward statements with longer verified proofs relative to their descriptions and substantially less overlap with the existing library.

3.4 Inference-Time Optimization for Theory Creation

In the previous section, we showed that we can successfully train a conjecturer to generate more interesting individual conjectures. But our ultimate goal is not just to generate individual interesting theorems, but to iteratively build a body of interesting theorems, where one round's discoveries become the premises for the next round. We showcase here how we can use our metric of interestingness to achieve this at inference-time, without updating the weights of the model. We evaluate an iterative theorem-discovery procedure, and we showcase that optimizing for interestingness gives us the strongest results across several metrics. We want to have a system that starts from a set of premises P_0, and then adds verified theorems to create an iterated increasing set of premises $P_0 \subset P_1 \subset \cdots \subset P_N$. At each iteration n, the conjecturer, here Claude 4.6, receives 20 premises, five of which are sampled from $P_{n-1} \backslash P_{n-2}$, and the rest from P_{n-1}. It generates 400 candidate statements. A Claude 4.6 based semantic filter removes equivalent conjectures and enforces diversity

across theorem families. The surviving statements are then proved independently with Claude Code in Lean. Successfully verified statements are ranked by the interestingness as computed from the ground-truth proof, and the ten highest scoring results are "promoted" and added to the next set of premises P_i. Appendix D compares this rule against retaining every verified statement, promoting ten statements uniformly at random, and promoting the ten statements with the longest verified proofs; interestingness pruning yields the highest running mean of promoted-statement interestingness and the highest quality, diversity, and "most interesting" judgments (Figure 7). We use the verified proof length here rather than V_θ because every candidate has already been proved at this stage, so the exact value of V is available; the learned estimator is needed only where proving every candidate is too time-consuming, as in the reinforcement-learning setting of Section 3.3. We showcase an experiment where P_0 is a set of premises from graph theory here in Figure 6; Appendix Figure 9 showcases the result of picking premises from Algebra, Measure / Probability Theory, and Number Theory.

The promotion rule at the end of each round is the only place where interestingness enters the loop, so we isolate its effect by holding the conjecturer, the semantic filter, and the proving agent fixed and varying only how the verified statements are pruned. We compare four rules: retaining every verified statement, promoting ten uniformly at random, promoting the ten statements with the longest verified proofs, and promoting the ten with the highest ground-truth interestingness. Each rule is run for 6 rounds from the same P_0, drawing on premises from graph theory, and we evaluate the resulting trajectories in two ways. We aim to quantify the quality and diversity of the samples from each of the runs, which we are able to do by using an LLM-as-a-judge. The judge (Claude 4.6) is never told which promotion rule produced the statements it evaluates. First, for each rule and round, it rates the cohort of ten promoted statements for quality and diversity on a 1–5 scale (Table 4). Second, it is shown four statements at a time, one from each rule, and ranks them by how mathematically interesting they are (Table 3). We also track the running mean of the verified interestingness of the sets P_n, and we find that interestingness pruning is the strongest promotion rule on each of these metrics, as displayed in Figure 7. Notably, pruning by raw proof length alone performs worse, indicating that the gain comes from the ratio defined in (6) rather than from simply favoring long proofs.

4 Discussion

Current LLM-based mathematics systems prove or solve human-posed problems; they do not decide which problems are worth posing. Our work takes a step toward changing this by defining a simple notion of interestingness. Crucially, our metric emerges from the structure of the proofs, and requires no human judgment of what constitutes interesting mathematics. Our results show that this metric is already sufficient to steer a language model toward non-trivial, out-of-distribution conjectures. Beyond individual conjectures, our iterative discovery procedure demonstrates that a system can build on its own outputs across rounds, with each round's most interesting theorems becoming premises for the next. Together, these results offer a proof of concept that interesting self-expanding mathematical libraries, grown without human intervention beyond the initial premises, are feasible.

We identify the following limitations with our specific methodology in this paper. Proof length in terms of lines of code can be a stylistic artifact, as automation tactics, such as `aesop` and `grind`, often trade proof length for runtime. Alternative notions such as the "Levin complexity" might provide a richer signal in light of this (Li & Vitányi, 2008). Our definition also isolates some aspects of mathematical value, and one may object that much of what makes a theorem interesting is that it connects previously unrelated objects or unifies disjoint subfields. Our definition of utility captures a weak form of this, but future work could try to quantify this notion more precisely.

Future work should aim towards extending our notions of discovery beyond just theorem statements to also allow for the creation of new and useful definitions, something we do not consider in this paper. Additionally, one could investigate training a model to estimate the proof difficulty in an online fashion, where as you expand your set of premises P_0, you are also updating your predictions of the proof length based on the ground truth outputs of your proving agent. Our results showed that LLMs are capable of creating a self-expanding mathematical library, which can have a high quality and diversity of theorems, but it is not yet clear if the new theorems are also useful for proving future results. Future work should look into the mechanisms for diversity and de-duplication that are required for long-running discovery systems.

AI DISCLOSURE

In this work, we used generative AI tools as core components of our method and evaluation. LLMs serve as the difficulty estimator, the theorem proposer, the proving agent (including a marginal-repair pass that may add hypotheses or narrowly weaken conclusions, Appendix C.2), the semantic filter that removes equivalent conjectures and enforces diversity (Section 3.4), the Mathlib-containment judge, the quality, diversity, and ranking judges whose scores are reported in Figure 7 and Tables 3–4, and the informalizations in Appendix C.3 and Figure 9 (Sections 2 and 3.3; prompts are found in Appendices C.1.4, C.2.3, and D). We have not used generative AI tools for research ideation or for generating experimental results outside of the pipeline described above, and the use of generative AI for human-subject research is not applicable to this work. Additionally, we used generative AI tools for assistance with writing experiment and plotting code and for polishing the language of the manuscript. We have reviewed all AI-assisted work, and all text was read and edited by the authors. We take responsibility for the final content of this work.

REFERENCES

Vitaly Aksenov, Eve Bodnia, Michael H. Freedman, and Michael Mulligan. Compression is all you need: Modeling mathematics. *arXiv preprint arXiv:2603.20396*, 2026. 5, 15

Boris Alexeev, Moe Putterman, Mehtaab Sawhney, Mark Sellke, and Gregory Valiant. Short proofs in combinatorics and number theory. *arXiv preprint arXiv:2603.29961*, 2026a. 1

Boris Alexeev, Moe Putterman, Mehtaab Sawhney, Mark Sellke, and Gregory Valiant. Short proofs in combinatorics, probability and number theory II. *arXiv preprint arXiv:2604.06609*, 2026b. 1

Noga Alon, Thomas F. Bloom, W. T. Gowers, Daniel Litt, Will Sawin, Arul Shankar, Jacob Tsimerman, Victor Wang, and Melanie Matchett Wood. Remarks on the disproof of the unit distance conjecture. *arXiv preprint arXiv:2605.20695*, 2026. 1

AlphaProof and AlphaGeometry teams. AI achieves silver-medal standard solving International Mathematical Olympiad problems. Google DeepMind Blog, July 2024. URL `https://deepmind.google/blog/ai-solves-imo-problems-at-silver-medal-level/`. 1, 15

Leni Aniva, Chuyue Sun, Brando Miranda, Clark Barrett, and Sanmi Koyejo. Pantograph: A machine-to-machine interaction interface for advanced theorem proving, high level reasoning, and data extraction in lean 4. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 104–123. Springer, 2025. 18

Anthropic. System Card: Claude Opus 4.6. System card, Anthropic, February 2026. URL `https://www-cdn.anthropic.com/0dd865075ad3132672ee0ab40b05a53f14cf5288.pdf`. 4

Scott Armstrong and Julia Kempe. Formalization of De Giorgi–Nash–Moser theory in Lean. *arXiv preprint arXiv:2604.05984*, 2026. 15

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*, 2023. 15

Maissam Barkeshli, Michael R. Douglas, and Michael H. Freedman. Artificial intelligence and the structure of mathematics. *arXiv preprint arXiv:2604.06107*, 2026. 1, 15

Yoshua Bengio and Nikolay Malkin. Machine learning and information theory concepts towards an AI mathematician. *Bulletin of the American Mathematical Society*, 61(3):457–469, 2024. doi: 10.1090/bull/1839. 15

Léonard Blier and Yann Ollivier. The description length of deep learning models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 15

Jorge Luis Borges. The library of babel. In Donald A. Yates and James E. Irby (eds.), *Labyrinths: Selected Stories and Other Writings*. New Directions, New York, 1962. 2

Xiaoyang Chen and Xiang Jiang. Moonshine: An autonomous mathematical research agent centered on conjecture generation. *arXiv preprint arXiv:2606.10806*, 2026. 1

Simon Colton, Alan Bundy, and Toby Walsh. On the notion of interestingness in automated mathematical discovery. *International Journal of Human-Computer Studies*, 53(3):351–375, 2000. 15

Randy Davila. Automated conjecturing with TxGraffiti. *arXiv preprint arXiv:2409.19379*, 2024. 15

Randy Davila, Boris Brimkov, and Ryan Pepper. In reverie together: Ten years of mathematical discovery with a machine collaborator. *arXiv preprint arXiv:2507.17780*, 2025. 15

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *International Conference on Automated Deduction (CADE)*, 2021. 1, 2, 3

Siemion Fajtlowicz. On conjectures of Graffiti. In *Annals of Discrete Mathematics*, volume 38, pp. 113–118. Elsevier, 1988. 15

Fabian Gloeckle, Jannis Limperg, Gabriel Synnaeve, and Amaury Hayat. Abel: Sample efficient online reinforcement learning for neural theorem proving. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024. 15

Fabian Gloeckle, Ahmad Rammal, Charles Arnal, Rémi Munos, Vivien Cabannes, Gabriel Synnaeve, and Amaury Hayat. Automatic textbook formalization. *arXiv preprint arXiv:2604.03071*, 2026. 15

Yichen Huang. Autonomous disproofs of the sum-product conjecture over $\mathbb{R}$ with GPT-5.5 Pro. *arXiv preprint arXiv:2607.20525*, 2026. 1

Andrei N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1), 1965. 15

Guillaume Lample, Marie-Anne Lachaux, Thibaut Lavril, Xavier Martinet, Amaury Hayat, Gabriel Ebner, Aurélien Rodriguez, and Timothée Lacroix. HyperTree proof search for neural theorem proving. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 15

Douglas B. Lenat. Automated theory formation in mathematics. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 833–842, 1977. 15

Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 3rd edition, 2008. 9, 15

Yong Lin et al. Goedel-Prover: A frontier model for open-source automated theorem proving. *arXiv preprint arXiv:2502.07640*, 2025. 15

Daniel Litt. Mathematics in the library of babel. Blog post, February 2026. URL `https://www.daniellitt.com/blog/2026/2/20/mathematics-in-the-library-of-babel`. 2

Miles Team. Miles: Enterprise-grade reinforcement learning for large-scale model post-training. `https://github.com/radixark/miles`, 2026. 4

Shubhra Mishra, Yuka Machino, Gabriel Poesia, Albert Jiang, Joy Hsu, Adrian Weller, Challenger Mishra, David Broman, Joshua B. Tenenbaum, Mateja Jamnik, Cedegao E. Zhang, and Katherine M. Collins. A matter of interest: Understanding interestingness of math problems in humans and language models. *arXiv preprint arXiv:2511.08548*, 2025. 2

Allen Newell and Herbert A. Simon. The logic theory machine—a complex information processing system. *IRE Transactions on Information Theory*, 2(3):61–79, 1956. doi: 10.1109/TIT.1956.1056797. 1

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. 1

OpenAI. GPT-5.5 System Card. Technical report, OpenAI, April 2026a. URL https://openai.com/index/gpt-5-5-system-card/. 4

OpenAI. Finite time blowup for Navier–Stokes. Technical report, OpenAI, 2026b. URL https://cdn.openai.com/pdf/32d9f210-8b73-45e0-91bc-82a30aef8a9a/navier-stokes.pdf. 1

Sang-il Oum. A proof of the cycle double cover conjecture by OpenAI: An exposition. *arXiv preprint arXiv:2607.16356*, 2026. 1

Gabriel Poesia, David Broman, Nick Haber, and Noah D. Goodman. Learning formal mathematics from intrinsic motivation. *arXiv preprint arXiv:2407.00695*, 2024. 2, 15

Henri Poincaré. *Science and Hypothesis*. Walter Scott Publishing Company, London, 1905. Translated by W. J. Greenstreet. 1

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020. 15

George Pólya. *Mathematical discovery: On understanding, learning, and teaching problem solving*. John Wiley & Sons, New York, combined ed. edition, 1981. ISBN 0-471-08975-3. 5

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026. URL https://qwen.ai/blog?id=qwen3.6-27b. 4, 7

Ahmad Rammal, Niket Patel, Fabian Gloeckle, Amaury Hayat, Julia Kempe, Rémi Munos, Charles Arnal, and Vivien Cabannes. Formalizing mathematics at scale. *arXiv preprint arXiv:2605.29955*, 2026. 15

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025. 1, 15

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 4

Ray J. Solomonoff. A formal theory of inductive inference, part I. *Information and Control*, 7(1), 1964. 15

Nat Sothanaphan. Resolution of Erdős problem #728: A writeup of Aristotle's Lean proof. *arXiv preprint arXiv:2601.07421*, 2026. 1

The mathlib Community. The Lean mathematical library. In *Certified Programs and Proofs (CPP)*, 2020. 1, 2, 3

William P. Thurston. Mathematical education, 2005. URL https://arxiv.org/abs/math/0503081. 6

George Tsoukalas, Rahul Saha, Amitayush Thakur, Sabrina Reguyal, and Swarat Chaudhuri. Learning interestingness in automated mathematical theory formation. *arXiv preprint arXiv:2511.14778*, 2025. 2, 15

A. M. Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, 45(1):161–228, 1939. doi: 10.1112/plms/s2-45.1.161. 1

Josef Urban. 130k lines of formal topology in two weeks: Simple and cheap autoformalization for everyone?, 2026. 15

Haiming Wang et al. Kimina-Prover Preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*, 2025. 15

Zichen Wang, Wanli Ma, Zhenyu Ming, Gong Zhang, Kun Yuan, and Zaiwen Wen. M2F: Automated formalization of mathematical literature at scale, 2026. 15

Eugene P. Wigner. The unreasonable effectiveness of mathematics in the natural sciences. *Communications on Pure and Applied Mathematics*, 13(1):1–14, 1960. doi: 10.1002/cpa.3160130102. 1

Yuhuai Wu, Albert Q. Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 15

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. DeepSeek-Prover: Advancing theorem proving in LLMs through large-scale synthetic data. *arXiv preprint arXiv:2405.14333*, 2024. 1, 15

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. In *International Conference on Learning Representations (ICLR)*, 2020. 15

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 1, 15

Huaiyuan Ying, Zijian Shi, Zhengying He, Bohan Gao, Chenyang Chen, Zhicheng Yu, Songrun Song, Qicheng Fan, Yinya Li, Lin Li, et al. Lean workbook: A large-scale Lean problem set formalized from natural language math problems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 15

CONTENTS

A Related Work **15**

B Training the Difficulty Model V_θ **15**

B.1 Data 15

B.1.1 Source corpus 15

B.1.2 Prompt 16

B.2 Model and Optimization 17

B.2.1 Training objective 17

B.2.2 Hyperparameters 17

B.3 Evaluation Protocols 17

C Training a Conjecturing Model **18**

C.1 Conjecturing Model Training 18

C.1.1 Data construction 18

C.1.2 Compilation, relevance, and triviality 18

C.1.3 Optimization configuration 19

C.1.4 Theorem-proposal prompt 19

C.2 Conjecturing Model Evaluation 19

C.2.1 Balanced verified cohort 19

C.2.2 Mathlib containment 20

C.2.3 Mathlib-containment rubric and judge prompt 20

C.3 Selected Generations 21

D Inference-Time Pruning Ablation **23**

D.1 Comparison across promotion policies 23

D.2 Judge prompts 24

E Additional Figures **27**

E.1 Cross-area interestingness matrix 27

E.2 Iterative discovery graphs 28

A Related Work

Language models for formal theorem proving. Language models can generate formal proof steps (Polu & Sutskever, 2020), with subsequent systems adding tree search and reinforcement learning (Lample et al., 2022; Gloeckle et al., 2024), library retrieval (Yang et al., 2023), synthetic data (Xin et al., 2024), subgoal decomposition, and test-time search (AlphaProof and AlphaGeometry teams, 2024; Ren et al., 2025; Lin et al., 2025; Wang et al., 2025). These methods primarily optimize completion of a fixed target under a particular prover and budget, whereas we aim to explore open-ended reasoning.

Autoformalization and formal libraries at scale. LLM autoformalization has progressed from statement-level translation (Wu et al., 2022) and paired datasets (Azerbayev et al., 2023; Ying et al., 2024) to large Lean developments from mathematical textbooks (Urban, 2026; Wang et al., 2026; Gloeckle et al., 2026). AutoformBot and Atlas further demonstrate coordination and verification across 26 books (Rammal et al., 2026). Recent work has also begun formalizing research-level results, such as the De Giorgi–Nash–Moser theory for elliptic PDEs (Armstrong & Kempe, 2026). These results establish the scalability of formal-library construction while exposing planning problems from missing infrastructure and long-range dependencies.

Automated mathematical discovery and conjecturing. AM and Graffiti pioneered automatic concept and conjecture generation (Lenat, 1977; Fajtlowicz, 1988), and interestingness itself became an explicit research problem (Colton et al., 2000). Graffiti's successors, notably TxGraffiti (Davila, 2024), have produced conjectures that led to multiple published papers over a decade of human–machine collaboration (Davila et al., 2025); however, these systems operate on graph invariants and their heuristic filters are intrinsically tied to that domain. Bengio & Malkin (2024) propose an information-theoretic view of mathematical interestingness, suggesting that valuable frameworks have small description length while being close to many provable statements; our definition can be seen as a computable, formal-library-grounded instantiation of this principle. More recent systems jointly learn conjecturing and proof in small axiomatic domains (Poesia et al., 2024) or learn interestingness objectives for theory formation (Tsoukalas et al., 2025). Proposals for autonomous discovery add novelty assessment, result selection, and closed-loop knowledge growth (Barkeshli et al., 2026). Our objective supplies such a signal at formal- library scale, grounding intrinsic interestingness in verified proof length and relating it to a theorem's extrinsic utility. This separates the ability to generate valid statements from the harder problem of selecting discoveries.

Description length and mathematical compression. Kolmogorov complexity measures shortest description length (Solomonoff, 1964; Kolmogorov, 1965; Li & Vitányi, 2008), while proof length and statement description length are distinct finite-corpus proxies. Resource-bounded information (Xu et al., 2020; Blier & Ollivier, 2018) and recent views of mathematics as a proof hypergraph compressed by useful abstractions (Barkeshli et al., 2026; Aksenov et al., 2026) motivate our library-grounded quantities. Interestingness compares proof and description length, while utility measures downstream proof compression; both are relative to the available mathematical context. They are computable, finite-corpus proxies rather than claims about uncomputable description complexity.

B Training the Difficulty Model V_θ

B.1 Data

B.1.1 Source corpus

From our extraction of MATHLIB, we get 113,547 declarations. For each declaration we record the fully qualified name, rendered type, and file path; the file-local definitions required to interpret the target and premises; the project-local theorem declarations referenced by the accepted proof; the source proof span and its non-blank physical line count; and the dependency metadata used to change which premises are visible.

From each theorem we build a group and are able to compute our reward within such a group. Fix a root theorem T, let P be the premises currently visible to the model, and let c be the materialized

computational-cost of that pair: the number of proof lines needed to derive T from P. Write D for the file-local definitions shipped with the prompt and $\mathrm{deps}(\cdot)$, for the extracted dependencies of a declaration. Sample a cited lemma uniformly from the theorem declarations among the visible premises,

$$L \sim \mathrm{Unif}\big(P\big)$$

Let ℓ be the proof cost of L. Now expand it: replacing L by its own dependencies gives the expanded premise set P^{-}, and dropping each of its premises independently gives the thinned premise set $P^{\cdot 8}$,

$$P^{-} = \big(P \setminus \{L\}\big) \cup \mathrm{deps}(L), \qquad P^{\cdot 8} = \{\, p \in P^{-} : u_p < 0.8 \,\}, \quad u_p \sim \mathrm{Unif}[0,1].$$

A proof from P^{-} can no longer cite L and must derive it, so it costs $c + \ell$. Writing D_{pre} for the definitions visible before the expansion and $D(L)$ for those of L alone, the edge emits six prompts (target | definitions, premises) together with their observed computational-cost labels.

Role	Prompt $\mapsto$ Label	Meaning
`base`	$(T \mid D,\, P^{-}) \mapsto c + \ell$	Predict the difficulty of proving T when L is not among the given premises; one would have to prove L along the way, so the answer grows by its ℓ lines.
`remaining_keep`	$(T \mid D,\, P^{-} \cup \{L\}) \mapsto c$	Predict the difficulty of proving T when L is handed back as a premise; it can be cited directly, so the answer returns to c.
`remaining_remove`	$(T \mid D_{\mathrm{pre}},\, P) \mapsto c$	Predict the difficulty of proving T from the premises available before L was expanded; the answer is again c, reached from a different set of premises.
`lemma_context`	$(L \mid D,\, P^{-}) \mapsto \ell$	Predict the difficulty of proving L itself from the premises the `base` row was given; this is exactly the ℓ lines that `base` had to add.
`lemma_native`	$(L \mid D(L),\, \mathrm{deps}(L)) \mapsto \ell$	Predict the difficulty of proving the same L from the premises it was actually proved from in MATHLIB; the answer must again be ℓ.
`base_drop`	$(T \mid D,\, P^{\cdot 8}) \mapsto \varnothing$	Predict the difficulty of proving T after each premise is dropped at random; it carries no label, since removing premises cannot make a proof shorter, and feeds only the one-sided penalty.

All six rows remain adjacent during reward construction, so that the reward can be evaluated within a group without any cross-batch bookkeeping. Roots, not individual prompt rows, determine the data split, so variants of one theorem never straddle the train/validation boundary.

For each root, generation starts from $P = \mathrm{deps}(T)$ and $c = \mathrm{lines}(T)$ and runs three deterministically seeded random expansion walks. After an edge is emitted, the walk continues from $(P^{-}, c + \ell)$. Each walk reservoir-samples at most three eligible edges, stops after 300 expansion attempts or once the visible premise set reaches 150 premises, and excludes prompts above 60,000 characters. The walks and reservoirs use seed 31; validation roots are randomly selected with probability $p = 0.01$.

The final dataset contains 18,318 training groups (109,908 prompts) and 923 validation groups (5,538 prompts). Grounded labels have mean 92.3, median 49, 90th percentile 245, and maximum 500 proof lines.

B.1.2 PROMPT

Every model receives the same instruction template. The frontier-model evaluations permit free-form reasoning but require the final integer after `####`; training uses identical semantic content.

```
You are given Lean definitions, premises, and a target theorem statement.

Estimate the proof difficulty: the total number of proof lines required
    to
derive the target theorem from these premises.
```

```
The definitions are file-local definitions that the statement and
    premises
depend on. Use them only to understand what the statement means -- do
    NOT
judge difficulty based on them, and do not count them as proof lines.

Pay attention to exactly which premises are available. Do not infer
difficulty from the number of premises alone.

Do not draft the proof. Just estimate the difficulty from the dependency
structure. Give a concise estimate, then put the final answer on its own
     line:

#### <integer>

Definitions:
[numbered Lean declarations]

Premises:
[numbered Lean hypotheses]

Target theorem:
theorem target : [rendered type] := by

What is the total proof difficulty (number of proof lines)?
```

Hypothesis names are normalized positionally to `h1`, `h2`, and so on; the target is renamed `target`; and any proof body after `:= by` is stripped. The model sees file-local definitions first, then premises, then the target.

B.2 Model and Optimization

B.2.1 Training objective

Write $m(x) = \log(1 + x)$, let v_{rj} be sampled completion j for role r, and let $\bar{v}_r$ be the mean of the parsed completions for that role within the batch; we abbreviate the six role means of `base`, `remaining_keep`, `remaining_remove`, `lemma_context`, `lemma_native`, and `base_drop` as $\bar{v}_{\text{base}}$, $\bar{v}_{\text{keep}}$, $\bar{v}_{\text{rem}}$, $\bar{v}_{\text{ctx}}$, $\bar{v}_{\text{nat}}$, and $\bar{v}_{\text{drop}}$. Algorithm 1 gives the per-completion reward used before GRPO normalization. Define the Bellman residual $a(x; y, z) = |m(x) - m(y+z)|$ and the agreement residual $q(x, y) = |m(x) - m(y)|$.

B.2.2 Hyperparameters

Parameter	Value	Parameter	Value
Base model	Qwen3.6-27B	Global batch	384 samples
Rollout batch	48 prompts	Samples per prompt	8
Rollout temperature	1.0	Context limit	131,072
Prompt limit	65,536	Response limit	8,192
Optimizer	Adam	Learning rate	10^{-6}, constant
β_1, β_2	0.9, 0.98	Weight decay	0.1
GRPO clip	0.20 / 0.28	KL coefficient	0.01
Steps	350	Policy loss	per token

B.3 Evaluation Protocols

Evaluation is performed with greedy decoding and an 8,192-token response cap, to match training. MAE, median absolute error, log-space R^2, and Spearman ρ are computed over all valid outputs. Every row from Claude and the trained model V_θ is successfully parsed, whereas GPT-5.5 produced 6 invalid outputs across all 4,615 prompts.

Algorithm 1 TRAININGREWARD — exact per-completion reward.

Input: completion $v = v_{rj}$ of role r; role means $\bar{v}_{\text{base}}$, $\bar{v}_{\text{keep}}$, $\bar{v}_{\text{rem}}$, $\bar{v}_{\text{ctx}}$, $\bar{v}_{\text{nat}}$, $\bar{v}_{\text{drop}}$; label c_r
Output: reward $\mathcal{R}_{rj}$

if v does not contain exactly one `</think>` followed by exactly one terminal `#### non-negative-integer` **then**
  return -2000
end if
$e_{\text{truth}} \leftarrow 0$ if $r = \texttt{base_drop}$, else $q(v, c_r)$
if $r = \texttt{base}$ **then**
  $e_{\text{add}} \leftarrow \frac{1}{4}\big[a(v; \bar{v}_{\text{keep}}, \bar{v}_{\text{ctx}}) + a(v; \bar{v}_{\text{keep}}, \bar{v}_{\text{nat}}) + a(v; \bar{v}_{\text{rem}}, \bar{v}_{\text{ctx}}) + a(v; \bar{v}_{\text{rem}}, \bar{v}_{\text{nat}})\big]$
  $e_{\text{drop}} \leftarrow [\, m(v) - m(\bar{v}_{\text{drop}})\,]_+$
else if $r = \texttt{remaining_keep}$ or $r = \texttt{remaining_remove}$ **then**
  $e_{\text{add}} \leftarrow \frac{1}{2}\big[a(\bar{v}_{\text{base}}; v, \bar{v}_{\text{ctx}}) + a(\bar{v}_{\text{base}}; v, \bar{v}_{\text{nat}})\big]; \quad e_{\text{drop}} \leftarrow 0$
else if $r = \texttt{lemma_context}$ **then**
  $e_{\text{add}} \leftarrow \frac{1}{3}\big[a(\bar{v}_{\text{base}}; \bar{v}_{\text{keep}}, v) + a(\bar{v}_{\text{base}}; \bar{v}_{\text{rem}}, v) + q(v, \bar{v}_{\text{nat}})\big]; \quad e_{\text{drop}} \leftarrow 0$
else if $r = \texttt{lemma_native}$ **then**
  $e_{\text{add}} \leftarrow \frac{1}{3}\big[a(\bar{v}_{\text{base}}; \bar{v}_{\text{keep}}, v) + a(\bar{v}_{\text{base}}; \bar{v}_{\text{rem}}, v) + q(v, \bar{v}_{\text{ctx}})\big]; \quad e_{\text{drop}} \leftarrow 0$
else ▷ $r = \texttt{base_drop}$
  $e_{\text{add}} \leftarrow 0; \quad e_{\text{drop}} \leftarrow [\, m(\bar{v}_{\text{base}}) - m(v)\,]_+$
end if
return $-0.50 e_{\text{truth}} - 0.35 e_{\text{add}} - 0.15 e_{\text{drop}}$

Calibration points in Figure 1 divide $[0, 450]$ proof lines into five equal-width 90-line bins. Within each observed-difficulty bin the marker gives the mean observed label against the mean prediction, and the shaded band spans the 25th–75th percentiles of predictions. Binning is used only for the plot on the left of Figure 1; every aggregate metric on the right is computed on unbinned rows.

C TRAINING A CONJECTURING MODEL

C.1 CONJECTURING MODEL TRAINING

C.1.1 DATA CONSTRUCTION

The theorem-proposal data are reconstructed from the `base_drop` rows of the corpus from B.1, and only the premises are used here. Premises are renamed positionally, and contexts with fewer than 16 premises or more than 12,288 tokens are removed. From this we randomly select 10,000 sets of premises for training and an additional 512 for validation, each set containing a median of 77 premises. The model sees definitions and premise statements but never the source theorem or its proof. Section C.1.4 gives the verbatim system and user templates.

C.1.2 COMPILATION, RELEVANCE, AND TRIVIALITY

A response must contain exactly one nonempty expression between «<STATEMENT»> and «<END»>. Pantograph (Aniva et al., 2025) compiles the expression as the type of a standalone theorem against the MATHLIB environment, to ensure it is syntactically valid. We only allow for the model to produce theorems and reject declarations, definitions, proof commands, `:=`, and `sorry`; parse failures receive -0.5 and expressions that fail compilations receive -0.25.

For a compiled type, the global definition graph computes $L(T \mid P)$ by subtracting the vocabulary exposed by the supplied premises. The proposal must share at least one non-generic identifier with a premise/definition context; failure receives -0.25. Direct premise restatements (as measured by token-level similarity) are marked trivial. Other candidates are tested for triviality using `assumption`, `rfl`, `simp`, `tauto`, and `simp_all`; a candidate closed by any one receives reward of 0. Only valid, relevant, nontrivial statements are sent to the V_θ LLM to judge the difficulty, and they then receive reward according to Equation 8.

C.1.3 OPTIMIZATION CONFIGURATION

Parameter	Value	Parameter	Value
Initialization	Qwen3.6-27B	Steps	75
Samples per prompt	8	Rollout temperature	1.0
Global / rollout batch	192 / 24	Context / prompt limit	12,800 / 12,288
Response limit	512	Planned updates	100
Optimizer	Adam	Learning rate	10^{-6}, constant
β_1, β_2	0.9, 0.98	Weight decay	0.1
GRPO clip	0.20/0.28	KL penalty	0
Save / evaluation interval	5 / 25	Policy loss	per token

C.1.4 THEOREM-PROPOSAL PROMPT

Bracketed fields such as `[NUMBERED PREMISES]` are populated mechanically with the corresponding Lean declarations; they are not additional instructions.

The system message is:

```
You are an expert Lean 4 and Mathlib theorem proposer. Answer immediately with the tagged
     theorem statement. Do not think aloud, explain, explore, or write a proof. Your first
     output must be <<<STATEMENT>>> and the whole response should stay under 256 tokens.
```

The user message is:

```
Given the Lean premises below, propose one new, interesting theorem statement that is a
     plausible logical consequence of them. The statement will be scored by estimated proof
     difficulty divided by its premise-relative definition length.

A strong theorem is nontrivial, concise, and uses multiple premises in a meaningful way. It
     must be a well-typed Lean proposition that could actually be proved from the available
     premises. Do not output False, an unsupported conjecture, a restatement of one premise,
     a vacuous implication, or a reflexive identity such as `x = x` or `P <-> P`. Use
     mathematical objects or operations supplied by at least two different premises. The
     output must be a standalone proposition: do not mention local aliases such as `h1` in
     the generated statement.

Prefer a short consequence obtained by instantiating and chaining the exact premises over a
     broad theorem with many freshly invented binders. Copy namespaces, binder types,
     typeclass assumptions, and function argument order exactly from the declarations below.
     Do not guess shorthand names or notation that is not shown. If you use a premise name
     such as `h2`, apply it only at its displayed signature.

Silently type-check the final expression before emitting it: fully apply maps that need
     arguments, give every new variable an explicit type, and use each operation only on
     values of its declared type (for example, do not add or subtract propositions). The
     output is a type, so do not use `let`, `by`, `:=`, holes, tactics, proof terms, or `#`
     cardinality notation. Do not print this check.

# Definitions for understanding the premise vocabulary
[NUMBERED DEFINITIONS, WHEN PRESENT]

# Available premise declarations ([POSITIONAL PREMISE NAMES])
[NUMBERED PREMISES]

# Output contract
Output only a single standalone Lean type expression between the tags. Do not reference the
     premise aliases above. If new variables are necessary, bind them with a leading `forall
     `/`forall`. Do not include a theorem name, `:= by`, proof, code fence, or commentary.

<<<STATEMENT>>>
<one concise Lean theorem statement>
<<<END>>>
```

C.2 CONJECTURING MODEL EVALUATION

C.2.1 BALANCED VERIFIED COHORT

The eight strata we select premises from are Algebra, Analysis, Number Theory, Measure/Probability, Geometry/Topology, Combinatorics, Category/Algebraic Geometry, and Foundations. For each group,

we sample theorems from each of the three models we evaluate (Claude, the trained conjecturing model, and the base model), and pass it to a Claude Code with Claude Opus 4.6 as a proof agent. We repeat until we have 20 distinct theorems from each area.

The proving agent first attempts the statement byte-for-byte. If that fails, a separate marginal-repair pass may add a necessary hypothesis, correct an index or bound, or narrowly weaken a conclusion. A repair is accepted only if it compiles, preserves the counts of $\forall, \exists, \leftrightarrow, \wedge, \vee, \neg$, passes anti-vacuity checks, and has a token-set Jaccard similarity at least 0.70, to disallow the statement becoming irrelevant or too easy. We recompute proof lines and conditional length from the final compiled declaration and compiled proof, so the reported score is ground-truth interestingness under the proving pipeline rather than the difficulty model's training reward.

C.2.2 MATHLIB CONTAINMENT

Claude Opus 4.6 at temperature 0 independently judges all 480 verified statements with proposer, area, and model hidden. Each prompt includes the final statement, verified proof body, and referenced MATHLIB declarations as non-exhaustive evidence. The proof scaffold is pinned to MATHLIB revision `905b95...`. Scores range from 1 (not contained) to 5 (an exact or trivially equivalent named declaration exists); score 4 denotes a few-line wrapper or immediate specialization. The judge must name a concrete declaration for score 5 and returns structured JSON. Figure 5 reports all of the recorded statements per model. The complete rubric and judge prompt are provided in Section C.2.3.

C.2.3 MATHLIB-CONTAINMENT RUBRIC AND JUDGE PROMPT

The five-level rubric is:

```
5 -- Fully contained. The exact statement (or a direct, trivially equivalent reformulation)
     exists in Mathlib as a named declaration. A user could cite it directly, or with a thin
     wrapper.

4 -- Substantially contained. The statement is provable in a few lines from existing Mathlib
     lemmas, or a strictly more general version is in Mathlib and the target is an immediate
     specialization.

3 -- Partially contained. Core ingredients (definitions and key supporting lemmas) exist in
     Mathlib, but the headline statement itself is not there and would require nontrivial
     assembly.

2 -- Minimally contained. Only background definitions or unrelated prerequisites are
     available; the substantive content is missing.

1 -- Not contained. Mathlib does not contain the statement or the necessary specialized
     definitions.

Operational guidance:
- Judge library containment, not mathematical validity, beauty, novelty, importance, or
     apparent difficulty.
- "A few lines" means a short Lean proof assembled directly from existing declarations, not
     merely a short informal mathematical argument.
- Use score 5 only when there is a concrete matching named declaration (possibly with
     symmetry, simplification, notation changes, or an immediate specialization). Identify
     that declaration whenever possible.
- A verified proof and its referenced Mathlib declarations are supplied as evidence. They are
      not exhaustive: a generated proof may overlook a more direct existing theorem, and
     proof length is not by itself dispositive.
- Do not infer a high score merely because the theorem is elementary. Conversely, do not
     infer a low score merely because its generated proof is long.
- If the exact declaration name is uncertain, report that uncertainty and avoid score 5
     unless the API match is otherwise unmistakable.
```

Each item is judged with:

```
You are an impartial expert judge of Lean 4 Mathlib library containment.

Apply the supplied rubric exactly. Estimate how much of the target theorem is already
     contained in Mathlib. The theorem has been verified, so do not judge whether it is true.
      Do not judge its mathematical novelty, importance, elegance, or difficulty.

The proof body and referenced declarations are supporting evidence, not an exhaustive Mathlib
      search. A generated proof can miss a direct existing theorem. Treat score 5 as
     requiring a concrete matching named declaration or an unmistakably direct API match.
```

```
    Distinguish a genuinely few-line Mathlib wrapper (score 4) from nontrivial assembly of
    ingredients (score 3).

<rubric>
[RUBRIC ABOVE]
</rubric>

<target_theorem>
[STATEMENT]
</target_theorem>

<verified_proof_body lines="[LINE COUNT]">
[PROOF BODY]
</verified_proof_body>

<referenced_mathlib_declarations>
[NAMES AND TYPES, OR NO RECOVERED LIST]
</referenced_mathlib_declarations>

Return exactly one JSON object and no Markdown or surrounding commentary:
{
  "score": <integer 1-5>,
  "matching_declaration": "<concrete Mathlib declaration for score 5, best candidate
     otherwise, or none>",
  "supporting_declarations": ["<zero or more important declaration names>"],
  "rationale": "<concise explanation tied to the rubric>",
  "confidence": "<high | medium | low>"
}
```

C.3 Selected Generations

Each item gives the final compiled Lean statement followed by a mathematical informalization.

```
∀ q : ℍ,
  q * q = -1 ↔ q.re = 0 ∧ ‖q‖ = 1
```

Quaternionic square roots of minus one. A real quaternion squares to -1 if and only if its real component is zero and its norm is one. Equivalently, the square roots of -1 are precisely the unit purely imaginary quaternions.

```
∀ z : UpperHalfPlane,
  ∃ n : ℕ, 0 < n ∧
    ‖Complex.exp
      (2 * ↑Real.pi * Complex.I * ↑z) ^ n‖ < 1 / 2
```

Decay of the upper-half-plane exponential. For every z in the complex upper half-plane, some positive power of $e^{2\pi i z}$ has absolute value less than $1/2$.

```
∀ (n : Type _) [Fintype n] [DecidableEq n]
    (R : Type _) [CommRing R]
    (A : Matrix n n R),
  A * A = 1 ↔
    A.det * A.det = 1 ∧
      A.adjugate = A.det • A
```

Involutory matrices and their adjugates. For a finite square matrix A over a commutative ring, $A^2 = I$ if and only if $(\det A)^2 = 1$ and $\operatorname{adj}(A) = (\det A)A$.

```
∀ κ : Cardinal,
  κ * κ = κ ↔
    κ = 0 ∨ κ = 1 ∨ Cardinal.aleph0 ≤ κ
```

Multiplicatively idempotent cardinals. A cardinal is unchanged by squaring exactly when it is 0, 1, or infinite (equivalently, at least $\aleph_0$).

```
∀ z : UpperHalfPlane,
  ∃ w : UpperHalfPlane,
    w = -(1 : ℂ) / (z : ℂ) ∧
      ∀ n : ℕ, n > 0 ↔
        (↑n : ℂ) * (z : ℂ) ≠ 0
```

Modular inversion in the upper half-plane. For every upper-half-plane point z, the modular inverse $w = -1/z$ is again in the upper half-plane. Moreover, for every natural number n, positivity of n is equivalent to the nonvanishing of the complex product nz.

```
∀ (R : Type _) [CommRing R],
  ∀ (W' : WeierstrassCurve.Jacobian R)
      (P : Fin 3 → R),
    W'.Equation (W'.neg P) ↔ W'.Equation P
```

Negation preserves the Jacobian equation. For a Jacobian Weierstrass model over any commutative ring, negating a projective point preserves the curve equation: $-P$ lies on the curve exactly when P does.

```
∀ (K : Type _) [Field K] (f : Polynomial K),
  Function.Injective
    (fun g : Polynomial K => g * f) ↔
      f ≠ 0
```

Injectivity of polynomial multiplication. Over a field, right multiplication by a polynomial f is injective on the polynomial ring if and only if f is nonzero.

```
∀ (K : Type _) [Field K] [NumberField K]
    (x : K),
  x ≠ 0 ↔
    ∀ w : NumberField.InfinitePlace K,
      (NumberField.InfinitePlace.mk
        (NumberField.InfinitePlace.embedding w)) x > 0
```

Positivity at every infinite place. An element of a number field is nonzero exactly when its value at every infinite place is strictly positive, where an infinite place evaluates through the absolute value associated with its embedding.

```
∀ (α : Type _) [GeneralizedBooleanAlgebra α]
    (a b : α),
  a = b ↔
    ∀ c : α, a \ c = b \ c ∧ c \ a = c \ b
```

Equality via relative complements. Two elements a, b of a generalized Boolean algebra are equal exactly when, for every c, removing c from them gives the same result and removing them from c also gives the same result.

```
∀ (V : Type _) [Finite V] (G : SimpleGraph V),
  G.edgeSet = ∅ ↔
    ∀ s : Set V, G.IsClique s ↔ s.ncard < 2
```

Empty graphs characterized by their cliques. A finite simple graph has no edges if and only if its cliques are precisely the vertex sets containing fewer than two vertices.

```
∀ (V : Type _) (G : SimpleGraph V) [Finite V],
  G.IsClique (Set.univ : Set V) ↔
    ∀ (M : Gᶜ.Subgraph),
```

```
      M.IsMatching ↔ M.verts = ∅
```

Completeness through matchings in the complement. A finite simple graph is complete if and only if a subgraph of its complement is a matching exactly when that subgraph has no vertices.

```
∀ n : ℕ, n > 0 ↔
  ∀ f : Polynomial ℝ, f.natDegree ≤ n - 1 →
    ((∀ k < n,
      Polynomial.eval
        (Polynomial.Chebyshev.node n k) f = 0) ↔
      f = 0)
```

Vanishing at Chebyshev nodes. A natural number n is positive exactly when every real polynomial of degree at most $n-1$ that vanishes at all n Chebyshev nodes is the zero polynomial (and the zero polynomial vanishes at those nodes).

```
∀ (C : Type _) [CategoryTheory.Category C]
    [CategoryTheory.Abelian C]
    (X Y : C) (f : X ⟶ Y),
  CategoryTheory.Mono f →
    (CategoryTheory.IsIso f ↔
      ∀ (Z : C) (g : Y ⟶ Z),
        (g = 0 ↔ f ≫ g = 0))
```

Detecting isomorphisms in an abelian category. Let $f : X \to Y$ be a monomorphism in an abelian category. Then f is an isomorphism exactly when precomposition with f detects zero morphisms out of Y: for every $g : Y \to Z$, one has $g = 0$ if and only if $g \circ f = 0$.

D Inference-Time Pruning Ablation

In this experiment, we aim to isolate the effect of the promotion rule in the recursive discovery loop of Section 3.4. All four trajectories we study start from the set of 80 premises P_0 from graph theory, and share the same Claude Opus 4.6 conjecturer model, and Claude Code based proving procedure. Each initial generation pass proposes 400 statements. Every prompt contains 20 premises: five sampled from the immediately preceding set $P_n \backslash P_{n-1}$ and 15 from the initial pool and older promoted rounds P_{n-1}.

These statements are proved independently in Lean. We compare four post-verification promotion rules: *no pruning* retains every verified novel statement; *random* retains ten uniformly at random; *proof length* retains the ten statements with the longest verified proofs; and *interestingness* retains the ten statements with the largest ratio

$$\widehat{I}_{\mathrm{ver}}(T \mid P) = 100\,\frac{\text{lines in the accepted proof of } T}{\text{characters in the normalized statement of } T}. \tag{9}$$

D.1 Comparison across promotion policies

All policies share the same initial premises and differ only in which verified statements are promoted. Table 1 compares the statements promoted over the ten rounds. Mean interestingness is computed within each round and then averaged across rounds, giving every round equal weight.

Promotion rule	Mean $\widehat{I}_{\mathrm{ver}}$	Median $\widehat{I}_{\mathrm{ver}}$	Mean proof lines
No pruning	1.313	1.114	3.80
Random	1.556	1.175	4.39
Proof length	3.259	2.480	**11.48**
Interestingness	**3.899**	**3.380**	7.41

Table 1: Quality of the promoted statements

Unsurprisingly, interestingness pruning produces the highest mean and median interestingness, while proof-length pruning produces the longest proofs. A direct advantage on promoted interestingness is partly expected because one policy selects on that quantity. We therefore separately evaluate every unique candidate successfully proved in Lean *before* the current round's promotion rule is applied. Table 2 reports their number, mean and median interestingness, and mean proof length across all ten rounds, using the same aggregation as Table 1.

Promotion rule	Mean $\widehat{I}_{\mathrm{ver}}$	Median $\widehat{I}_{\mathrm{ver}}$	Mean proof lines
No pruning	1.313	1.114	3.80
Random	1.461	1.271	3.74
Proof length	1.694	1.167	**5.46**
Interestingness	**2.332**	**1.754**	4.56

Table 2: Verified candidates before selection.

The judge sees four statements at a time, one from each policy, and ranks them by mathematical interestingness. Policy names, proofs, and scores are hidden. We evaluate 100 such groups, and show the results below.

Promotion rule	Ranked first	Ranked in top two	Mean rank ↓
No pruning	13%	34%	2.89
Random	8%	29%	2.95
Proof length	19%	64%	2.36
Interestingness	**60%**	**73%**	**1.80**

Table 3: Direct policy-blinded judgment of mathematical interestingness.

The judge also evaluates ten groups of ten statements promoted by each policy in each round as a group. Policy names and scores are hidden. The judge rates the group's overall quality and diversity and identifies repeated theorem families. For no pruning, which retains more than ten statements, we select ten at random.

Promotion rule	Quality (1–5)	Diversity (1–5)	Families per ten	Redundant
No pruning	2.50 ± 0.53	1.30 ± 0.48	2.60 ± 0.97	$50.0\% \pm 26.7\%$
Random	2.40 ± 0.52	1.20 ± 0.42	2.90 ± 0.99	$53.0\% \pm 18.9\%$
Proof length	2.60 ± 0.52	1.30 ± 0.48	2.90 ± 0.74	$50.0\% \pm 18.3\%$
Interestingness	$\mathbf{3.40 \pm 0.52}$	$\mathbf{2.10 \pm 0.32}$	$\mathbf{3.50 \pm 0.71}$	$\mathbf{28.0\% \pm 7.9\%}$

Table 4: Policy-blinded cohort judgment. Entries are mean ± sample standard deviation across the ten round-level cohorts for each promotion rule.

D.2 JUDGE PROMPTS

The complete judge instruction is deliberately short and uses a broad notion of what a mathematician might appreciate:

```
You are a mathematician comparing four formally verified theorem
    statements.

Rank them from most to least mathematically interesting. Use a broad
    notion of interestingness: a mathematician might appreciate a
    statement because it is nontrivial, surprising, elegant, conceptually
     illuminating, potentially useful or reusable, connects ideas, or
    opens further questions. Judge the mathematical content, not notation
    , verbosity, statement length, or guessed proof length.

[A]
```

```
<THEOREM>

[B]
<THEOREM>

[C]
<THEOREM>

[D]
<THEOREM>

Return exactly one JSON object and no Markdown:
{"ranking":["A","B","C","D"],"rationale":"one or two sentences","
    confidence":"high|medium|low"}
```

The complete cohort-judge prompt is:

```
You are an impartial expert judge of a collection of ten verified Lean 4
     theorem statements.

The statements are all correct. Judge the collection itself, without
    guessing its source or generation method.

Rate two dimensions from 1 to 5.

MATHEMATICAL QUALITY
5: Mostly nontrivial, coherent, potentially reusable results that
    express substantive mathematical content.
4: Generally meaningful results with some routine or overly specialized
    items.
3: Mixed quality; several useful statements but substantial routine
    conjunctions, wrappers, or narrow variants.
2: Mostly shallow, over-specialized, or mechanically assembled
    statements.
1: Almost entirely trivial, incoherent, vacuous, or unhelpful statements.


SEMANTIC DIVERSITY
5: Many genuinely distinct theorem families or mathematical ideas, with
    little redundancy.
4: Several distinct families and only limited repetition.
3: A meaningful mixture, but multiple statements repeat the same core
    pattern.
2: Dominated by one or two narrow families with superficial variations.
1: Nearly all statements are equivalent, nested variants, or routine
    repackagings.

Group statements by their substantive conclusion and proof idea, not
    merely surface syntax. A stronger/weaker variant or conjunction of
    the same core facts should normally stay in the same family.

<statements>
[1]
<THEOREM>
...
[10]
<THEOREM>
</statements>

Return exactly one JSON object and no Markdown:
{
  "quality_score": <integer 1-5>,
  "diversity_score": <integer 1-5>,
  "distinct_families": <integer 1-10>,
  "redundant_statement_indices": [<zero or more integers 1-10>],
```

```
  "family_assignments": [{"indices": [<integers>], "family": "<short
    label>"}],
  "rationale": "<concise explanation>",
  "confidence": "<high | medium | low>"
}
```

E ADDITIONAL FIGURES

E.1 CROSS-AREA INTERESTINGNESS MATRIX

The displayed matrix uses ten targets from each of 12 areas. Starting from one area-conditioned premise frontier per column, we apply three synchronous expansion rounds: every current extracted theorem premise is replaced by all original theorem premises used in its proof, unavailable declarations remain terminals, and each resulting frontier is deduplicated. Fresh V_θ predictions are obtained for all 1,440 target–condition pairs. Each target's score is divided by its same-area score before cell medians are formed. Only 28 off-diagonal 95% confidence intervals exclude one. Across the 132 off-diagonal cell medians, the median is 0.520 and 73.5% are below one. We see here that, as we might expect, a definition-heavy field such as measure theory becomes much less interesting when it is not given premises from analytic areas. We find also that some areas, such as algebra, become more interesting when supplied with premises from other fields.

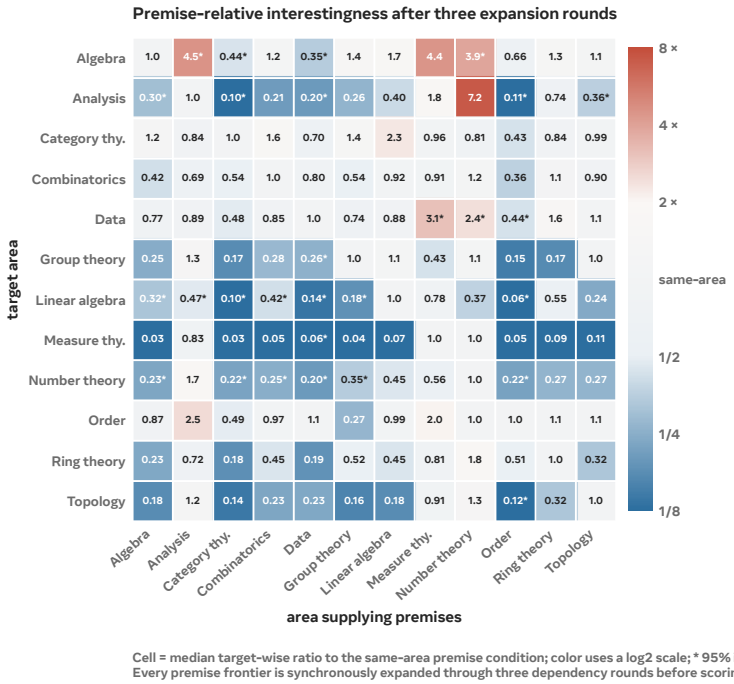
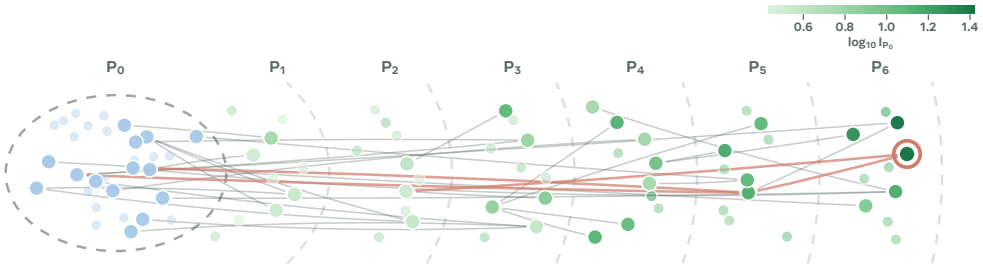


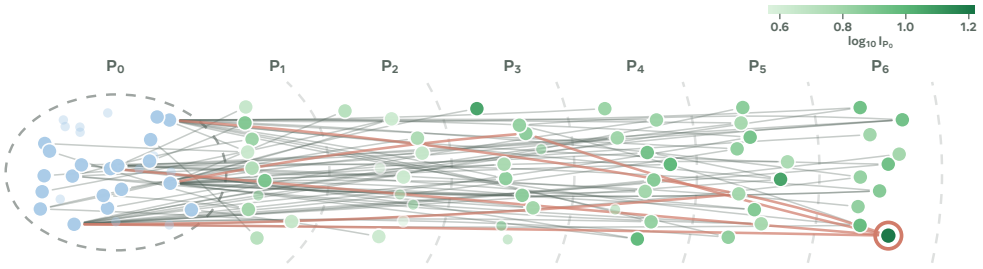
Figure 8: Full cross-area interestingness matrix. Each cell is the median target-wise ratio for the row area under premises from the column area versus the same-area premise condition. All premise frontiers are synchronously expanded through three rounds before new predictions are obtained. We select ten targets per area (120 total); four row areas contribute nine ratios after each excludes one non-positive same-area score. Asterisks mark 95% confidence intervals that exclude one.

E.2 ITERATIVE DISCOVERY GRAPHS

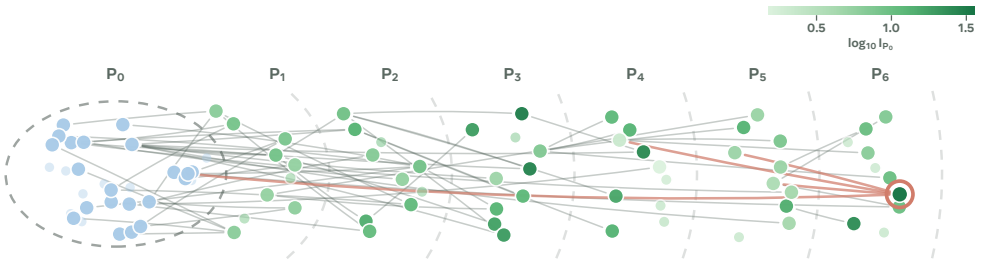
Figure 9 provides more iterative discovery graphs.



(a) Algebra. For monic q and r, reducing pr modulo qr equals reducing p modulo q and then multiplying by r.



(b) Measure/probability. If μ is absolutely continuous with respect to ν, then under the stated measurability conditions, the sum of the pushforwards of $c\mu$ and μ is absolutely continuous with respect to $(c+1)$ times the pushforward of ν.



(c) Number theory. Applying a real invertible 2×2 matrix to a nonreal complex number and then applying its inverse restores the imaginary part.

Figure 9: Iterative forward-discovery graphs through P_6. Each panel shows 30 of the 240 initial premises in P_0, always retaining every initial premise used by a displayed proof. Gray lines record explicit dependencies in accepted Lean proofs; coral marks the ancestry of the theorem with the highest total P_0-relative interestingness among the statements introduced in P_6. Green encodes $\log_{10} I_{P_0}$ with a separate scale in each panel. Each subcaption reports the selected statement.