FORMALPROOFBENCH：模型能否编写经过形式化验证的研究生数学证明?

Nikil Ravi[1] **Kexing Ying**[2] **Vasilii Nesterov**[3] **Rayan Krishnan**[1] **Elif Uskuplu**[4] **Bingyu Xia**[5] **Janitha Aswedige**[5] **Langston Nashold**[1]

[1]Vals AI [2]洛桑联邦理工学院（EPFL）[3]莫斯科物理技术学院 [4]印第安纳大学伯明顿分校[5]独立研究者

arXiv:2603.26996v1 [cs.AI] 27 Mar 2026

摘要

我们介绍了 **FormalProofBench**，这是一个私有[1] 基准测试，旨在评估人工智能模型能否生成研究生水平的形式化验证的数学证明。每项任务都将一个自然语言问题与一个 Lean 4 形式化陈述配对，模型必须输出一个能够被 Lean 4 检查器接受的 Lean 证明。FormalProofBench 面向高年级本科和研究生数学，问题取自资格考试和标准教科书，涵盖分析学、代数学、概率论和逻辑学等主题。我们使用智能体评测框架评估了一系列前沿模型，发现表现最佳的基础模型准确率达到 33.5%，此后性能迅速下降。除准确率数据外，我们还对工具使用、失败模式、成本和延迟进行了实证分析，从而全面评估前沿模型进行形式化定理证明的能力。

1 引言

大语言模型在数学推理方面取得了显著进展。近期系统在竞赛类基准测试中获得了高分，例如 MATH（Hendrycks 等人，2021）、GSM8K（Cobbe 等人，2021）、国际数学奥林匹克竞赛，以及难度更高的数学基准测试 FrontierMath（Besiroglu 等人，2024）。这些进展得益于在数学语料上进行预训练、通过人类反馈和自动反馈进行强化学习后训练，以及推理时推理等方面的进步。

然而，自然语言解答可能掩盖细微的逻辑错误，而人工评分者和基于大语言模型的评分者都难以审查这些错误。模型可能给出听起来似是而非的论证，却遗漏关键情形或援引并不存在的引理；甚至专家审阅者也可能忽略这些错误。表面上的数学能力与表面 经验证 的数学能力之间存在差距，因此有必要将形式化验证用作评估方法：证明要么通过类型检查，要么无法通过。

与此同时，数学界对交互式定理证明器的采用日益广泛。Lean 4（de Moura 等，2021）及其数学库 Mathlib（mathlib Community，2020）迅速发展，而 Liquid Tensor Experiment（Commelin 与 Topaz，2023）等形式化工作，以及仍在进行中的费马大定理形式化工作（Buzzard 与 Taylor，2025），表明现代证明助手能够扩展至深入的研究级数学。与此同时，近期有多项研究探索了人工智能与形式数学的交集，包括谷歌 DeepMind 的 AlphaProof（Hubert 等，2025），以及用于自动形式化的 Aristotle 系统（Achim 等，2025）。这些进展凸显了严格基准测试的重要性与迫切需求，用以衡量语言模型能否为非平凡数学问题生成机器可检验证明。

[1]T该基准测试保持私有，以防止数据污染。为提高透明度，我们提供示例证明问题。见附录 A.1；此外，评估提示和所有测试框架代码均可通过此 https 网址获取。

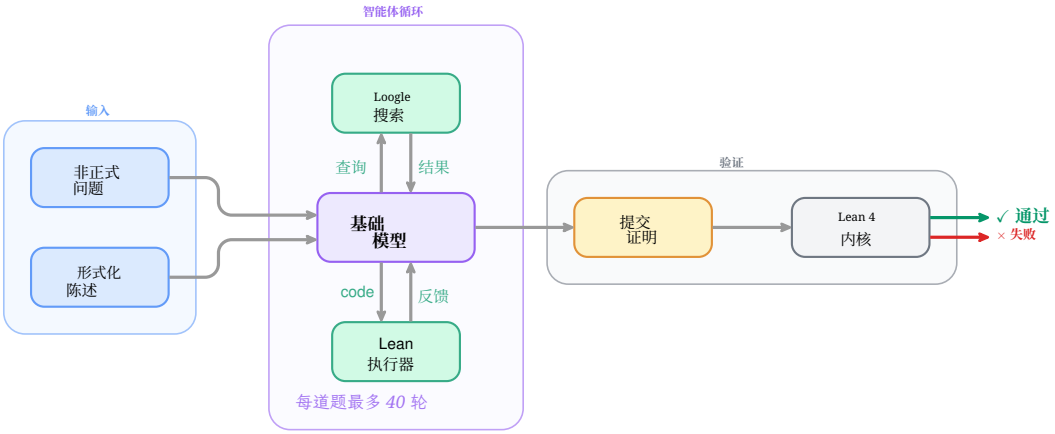


图 1：**FormalProofBench 的评估流程**。模型会获得非形式化问题陈述以及形式化陈述（+ 必要的头文件和上下文），并有 40 轮时间（每轮可能涉及多次工具调用和推理）来提交证明。

我们推出 **FormalProofBench**，这是一个私有基准测试，旨在评估模型在高年级本科和研究生水平上的形式化定理证明能力。每项任务都将自然语言问题陈述与 Lean 4 中对应的形式化陈述配对；模型必须生成能被 Lean 内核接受的证明项。问题取自资格考试和标准教材，涵盖测度论、实分析与泛函分析、代数学与交换代数、代数几何、数论、概率论与随机分析，以及逻辑学。所有形式化工作均由经验丰富的 Lean 用户（包括博士生和领域专家）完成，并经过二次审查流程，以避免错误形式化。

与非形式化数学基准不同，FormalProofBench 提供二元且完全自动化的正确性信号：证明要么能够编译通过，要么不能。这消除了由模型或人工评估解答正确性的必要，并确保不会因看似合理但存在缺陷的推理而获得分数。

我们在智能体框架中评估了一系列前沿模型，为每个模型提供搜索 Mathlib、执行 Lean 代码和提交证明的工具，每道题最多允许进行 40 轮交互。

主要贡献与要点。

1. 我们提出了一项全新的、具有挑战性、贴近现实、可确定性评估且未受污染的语言模型性能基准测试。2. 我们使用 200 道经过仔细形式化的问题，对前沿模型在**研究生水平**的形式化定理证明任务上的表现进行了详细评估。3. 我们分析了模型在**可使用搜索工具和代码执行工具的真实多轮智能体框架**中产生的**成本和延迟**，其中模型最多可进行 40 轮交互以提交解答。4. 我们发现，**Anthropic 的 Claude Opus 4.5 (Thinking)**表现最佳，准确率达到 33.5%，此后性能迅速下降。5. 工具使用模式表明，**频繁通过代码执行进行迭代**与更强的性能相关。

论文结构。 第2节回顾数学基准测试和形式化定理证明方面的相关工作。第3节介绍基准构建，包括问题选择、形式化和质量保证。第4节详细介绍评估框架和工具接口。第5节展示结果并进行分析。第6节讨论局限性和未来方向。

2 相关工作

非形式数学基准测试 自然语言数学（我们称之为“非形式数学”）已被广泛用于评估大语言模型并指导其训练。两个广泛采用的基准是 MATH（Hendrycks 等人，2021），其中包含 12,500 道竞赛数学问题；另一个是美国数学邀请赛（AIME），这是一个广受欢迎的基准，用于衡量模型解决美国数学邀请赛问题的能力。近年来，IMProofBench 和 FrontierMath 等基准开始评估模型处理难度更高的研究级任务的能力。

形式数学基准测试 若干基准测试从不同难度层级评估神经定理证明器。MiniF2F（Zheng 等人，2022）是一个跨系统基准，包含以多个证明助手形式化的奥林匹克竞赛级问题。PutnamBench（Tsoukalas 等人，2024年）将其扩展至大学竞赛数学，提供了以 Lean、Coq 和 Isabelle 编写的普特南竞赛题。ProofNet（Azerbayev 等人，2023）聚焦于 Lean 3 中的本科级定理证明和自动形式化。FormalMATH（Yu 等人，2025年）提供了一个规模更大的基准测试，也涵盖本科数学。

领域专用基准测试针对特定数学子领域。IneqComp（Zhao 等人，2025年）通过数学不等式测试组合推理能力。CombiBench（Liu 等人，2025年）聚焦组合数学，包含 100 道以 Lean 4 形式化的竞赛题。MATP-Bench（He 等人，2025年）在形式化环境中评估模型的多模态推理能力。

形式化定理证明 在方法方面，使用大语言模型进行自动形式化（Wu 等人，2022）表明，大语言模型能够将自然语言数学内容转换为形式化规范。GoedelProver（Lin 等人，2025年）在此基础上，通过对自动形式化的证明进行迭代训练，在 MiniF2F 和 PutnamBench 上取得了出色的结果。LeanDojo v2（Hsiang 等人，2025年）提供了一个用于 Lean 4 中 AI 辅助定理证明的端到端框架，将代码库提取、大语言模型微调和证明搜索整合在一起。Lean Workbook（Ying 等人，2025年）提供了约 57,000 组形式与非形式问题对，其中包括通过迭代式翻译流程生成的国际数学奥林匹克题目。SorryDB（Letson 等人，2026）为公开 Lean 代码库中的不完整证明（“sorries”）建立索引，作为一项基于公开可用的未解决 sorry 的实时基准测试。

FormalProofBench 旨在评估模型解决高年级本科或研究生水平任务时的形式化定理证明能力，而非解决高中水平的问题。此外，我们使用智能体框架评估模型，而不是评估它们能否在单次前向传播中生成证明。

3 基准测试概述

FormalProofBench 中的每项任务都包含：(i) 一个自然语言表述的数学问题，以及 (ii) 对应的 Lean 4 形式化陈述。给定这两项内容后，模型必须构造出一个能通过证明检查器验证的 Lean 证明。

3.1 数据整理

问题由六位形式化人员筛选并形式化，他们是博士生或具有 Lean4 代码编写经验的专家。他们都拥有丰富的 Lean 经验，或曾向 Mathlib 提交并合并贡献，或参与过此前的大规模形式化工作；此外，他们的数学兴趣各不相同，使我们能够形式化来自不同领域的问题。我们收集问题时遵循以下标准：

- 需要超出奥赛风格／高中风格题目范围的知识的问题。
- 能够使用当前版本的 Mathlib 形式化，且证明也能够使用当前版本的 Mathlib 编写的问题。
- 这些问题足以挑战截至2025 年末具备合理能力的模型，同时仍与该数学领域的研究相关。

- 涵盖多种数学领域。

任务示例请见列表 1；该问题及其他问题的完整模型回答见附录 A.1。

问题来源与领域 这些问题取自高年级本科及研究生资料，包括资格考试和标准教科书。它们涵盖现代数学的广泛领域，包括测度论、实分析与泛函分析、代数学与交换代数学、代数几何、数论、概率论与随机分析，以及逻辑学和模型论。表 1 展示了这 200 个基准问题在这些领域中的分布。

学科领域	**数量**
实分析	56
概率论 / 随机分析	34
泛函分析	26
数论	23
逻辑学 / 模型论	21
代数学	18
代数几何	15
线性代数	7
总计	**200**

表 1：FormalProofBench 问题按学科领域的分类。

数据质量与错误形式化检查 虽然 Lean 4 可以检查一个证明是否确实证明了某个陈述，但形式化数学基准测试中的一个常见问题是，问题陈述存在错误形式化，原因可能是定义不一致，或未能正确转换为 Lean 4。换句话说，无法保证形式化陈述忠实表达了非形式化陈述的意图。此外，众所周知，形式化基准测试往往存在问题，包括问题被错误形式化以及遗漏附加条件（Ammanamanchi & Bhat，2025年）。
考虑到这些问题，我们的数据收集流程纳入了第二阶段的审核，由人工审核并修订问题陈述。第二阶段审核采用 GitHub 上的标准代码审查流程进行，由一名或多名非问题原始贡献者审核非形式化数学陈述、自然语言证明和形式化陈述。

搜索工具的必要性基准测试中的一些问题，其解法依赖于近期才加入 Mathlib 的定理。虽然模型原则上可以重新证明所需的任何引理，但我们仍纳入这些问题，以测试现实世界形式化中的一项关键能力：可靠地查找相关定理。

3.2 评测设置

评测中一个常见的问题是，不同模型的设置各不相同，例如推理力度、温度、top_p、推理服务提供方等。我们使用各模型提供方的默认采样设置，并采用原生模型提供方而非第三方推理服务提供方，从而统一评测设置。

4 评估框架与工具

我们向模型提供自然语言的问题描述及其形式化版本，并附上必要的头文件和上下文。我们还提供以下工具：

- lean_loogle：模型可通过 https://loogle.lean-lang.org 使用此工具搜索 Mathlib 中的定理和定义。
- lean_run_code：模型可以使用此工具执行任意 Lean 代码并获取反馈，包括执行成功或失败、错误描述、未解决的目标、类型不匹配、警告、sorry、语法错误、缺失的定义/引理等。

```
导入Mathlib

打开拓扑空间 滤子 测度论 概率论
    函数
打开 scoped 非负实数 ENNReal 测度论 概率论
    拓扑学
命名空间 测度论

变量 {Omega: 类型*}  [欧米伽可测空间]

/-- 证明: 在
可测空间 (Omega, F) 上, 具有以下"连续性"性质的
Bn 属于 F, Bn → 为空集, mu(Bn) < inf => mu(Bn) → 0 必须是
可列可加的, 前提是 mu(Omega) < inf。 -/
定理Dembo_1_1_15 {mu : 集合 欧米伽 → 实数}
    (hAdd: 对于所有AB , 可测集 A →可测集 B →
      不相交 A B → 测度 (A \cup B) = 测度 A + 测度 B)
    (hCont : 对于所有 B : 自然数 → 集合 欧米伽,
      (对于所有 n , 可测集 (B n)) →
      反单调 B →(交集 n, B n) = 空集 →
      趋于 (函数 n => 测度 (B n)) 趋于正无穷 (𝓝 0))
    {A : 自然数 → 集合 欧米伽} (hA : 对所有 n,        可测集 (A n))
    (hDisj : 两两 (不相交 on A)) :
    测度 (Union n,      一个 n) = 对 n 求和, 测度 (A n) := 根据
  抱歉
end 测度论
```

列表 1：基准测试中的一段形式化示例，摘自 Dembo 的概率论讲义 Dembo (2024年)（练习 1.1.5）。

- 提交_证明：用于表示最终提交的证明，以 by 开头。

为了让模型在提交前能够反复完善解答，每道题最多允许 40 轮。只有当模型调用 submit_proof 并提交通过 Lean 4 内核的证明时，该解答才被视为有效。我们使用 Lean **v4.25.2** 和 Mathlib（提交版本为 c98ae54af00eaefe79c51b2b278361ca94e59bfb）。此外，我们会拒绝任何包含 axiom 或 local_instance 的尝试，因为这会规避形式化验证。

```
系统_提示词 ="""
你是一位 Lean 4 定理证明专家，同时也是研究生水平
数学领域的专家。你将收到一个研究生水平的数学定理陈述
及其 Lean 4 形式化表达。你必须输出 Lean 4 证明（你的输出
只能包含最终证明；也就是仅保留以“by”开头的部分）。
请注意定理陈述的形式化方式；其中可能存在
一些特定的细微之处。这对
理解定理陈述并对其进行形式化证明至关重要。
你将可以使用一组工具来帮助你证明定理。你
可以在需要时借助这些工具；请明智地判断何时以及如何使用它们。
你可以在预算和轮次限制内随意使用这些工具，
但完成后必须调用 submit_proof 提交最终证明（
即以 'by' 开头的部分）。
请使用有效且正确的 Lean 4 语法；如有疑问，可以借助这些工具。
Lean 4 语法要点：
- 引理名称使用驼峰命名法（例如 Nat.dvd_mul，而不是 nat.dvd_mul）
- 使用 ';' 串联策略                    而不是 ','
- 重写需要加括号：rw [h] 或 rw [step1, step2]
- 合取使用 'constructor'，而不是 'split'
- 环公理使用 'ring'，                    线性算术使用 'linarith'
- 存在量词见证：使用 1                  , 2, 3 不要使用 [1, 2, 3]
- 策略前不要缩进
- 在最终提交中，绝不要使用 'sorry'，即使是在注释中
  或在最后一步中。
"""
```

图 2：用于所有模型评估的系统提示。

```
用户_提示_模板 = """请在 Lean 4 中证明以下定理。
自然语言陈述：
{natural}
形式化陈述：
```lean
{header}
{formal}
```
{tool_guidance}
请提供一个以“by”开头的证明。需要提交最终证明时，请调用 submit_proof
当你想提交最终证明时。
```

图 3：用户提示模板。大括号中的变量会根据具体问题填入。工具指导内容通过 {tool_guidance} 附加。完整的工具指导提示见图 6。

5 结果

我们没有评估多个较小的开源模型，而是选择评估实力更强的前沿基础模型，旨在对当今能力最强的模型进行基准测试。表 2 总结了 OpenAI、Anthropic、xAI、智谱 AI、谷歌和 DeepSeek 模型的整体结果。

Rank	模型	准确率（%）	每次测试成本（美元）	每次测试延迟（秒）
1	Claude Opus 4.5 (Thinking)	33.50	1.34	4036.78
2	Gemini 3 Pro (11/25)	18.50	1.26	9966.43
3	Claude Sonnet 4.5（思考）	18.00	0.84	11186.79
4	GPT 5	17.00	0.89	5017.28
5	Gemini 3 Flash (12/25)	15.00	0.33	1305.37
6	GPT 5.2	13.50	1.10	1459.24
7	GPT 5 Nano	12.50	0.05	3994.69
8	GPT 5.1CodexMax	12.00	0.55	2552.82
9	DeepSeek V3.2（非思考模式）	8.00	0.12	3715.92
10	GPT 5 Mini	7.00	0.21	2805.07
11	GLM 4.7	4.50	0.15	3403.63
12	Grok 4.1 Fast（推理模式）	4.00	0.10	1455.68
13	DeepSeek V3.2（思考模式）	3.00	0.13	3808.62

表 2：FormalProofBench 总体排行榜结果（准确率（%）、每道测试题的成本和延迟）。

5.1 工具使用分析

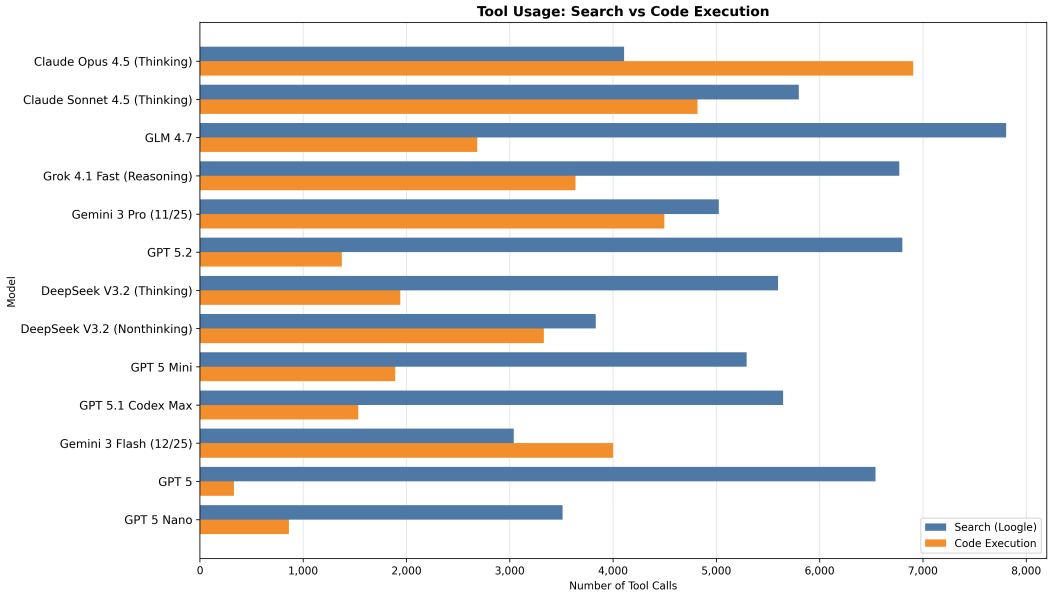


图4：各模型在全部 200 道问题上的工具调用总数（搜索与代码执行）。不同模型的工具使用策略差异很大。

一种常见的失败模式是过度搜索：模型有时会陷入反复调用 lean_loogle 工具的循环，试图寻找并不存在的引理。相比之下，能力更强的模型往往会较早开始执行代码，并在证明过程中通过搜索填补空缺。我们观察到，代码执行反馈频率越高，整体成功率也越高。从定性角度看，这是因为倾向于执行代码的模型能获得更多反馈，因此可以不断迭代，逐步得出正确解答；只有在确有需要时才使用搜索工具（见图4）。

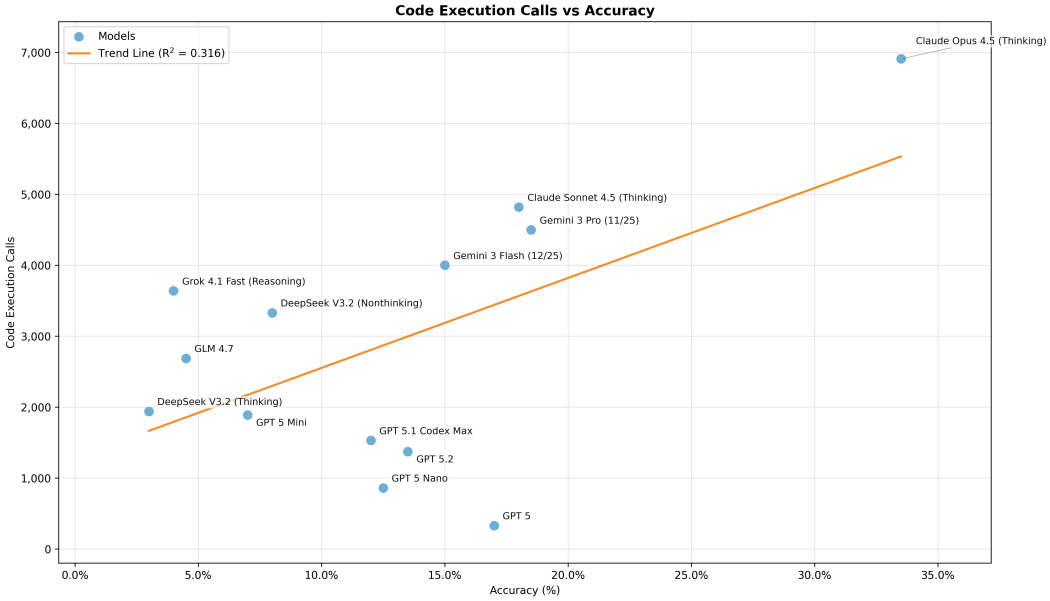


图 5：我们观察到，各模型的代码执行调用次数与准确率之间存在正相关，但相关性较弱。值得注意的离群值包括 GPT 5，它只进行很少的代码执行调用，却取得了相对较高的准确率；以及 Grok 4.1 Fast，它执行了大量代码，但准确率较低。

5.2 延迟与成本

除了衡量准确率之外，衡量各模型达到相应准确率所需的成本和延迟也很重要。延迟按整个查询执行过程（包括重试）的实际耗时计算，成本则根据令牌用量和各模型特定的定价规则计算；成本计算中也纳入了缓存因素。每项测试的平均成本约为 10 美分至 1 美元以上。表 2 中的延迟数据表明，在此基准测试中，模型尝试得出最终提交结果时，每项测试的平均耗时往往超过一小时。

6 结论与未来工作

我们推出了 FormalProofBench，这是一个用于评估人工智能模型能否生成研究生水平的形式化验证的数学证明的基准测试。我们对前沿模型的评估显示，表现最佳的系统 Claude Opus 4.5 (Thinking) 的准确率为 33.5%，此后表现迅速下滑。通过对工具使用模式进行详细分析，我们发现，通过代码执行频繁迭代与成功密切相关，而不执行代码、仅进行过度搜索则往往表明进展路径难以成功。

一个兼具多项理想特征的基准测试。 FormalProofBench 在人工智能评估领域占据独特地位。近期的基准测试（如 FrontierMath）优先纳入具有现实意义的难题。FormalProofBench 不仅如此，还具备几项难以兼顾的额外优势：（i）通过 Lean 4 内核进行确定性评估，无需模型评分或人工评分的正确性检查；（ii）抗污染能力强，因为我们创建的特定 Lean 形式化内容不太可能出现在训练数据中；（iii）评估成本实用，每道题的评估只需几分钟到几小时，而非数天；（iv）可求解性，因为我们选择的问题能够使用当前的 Mathlib 在 Lean 中正确陈述并求解。这些优势兼备，使 FormalProofBench 对追踪模型不断进步的情况尤其有价值。

对数学实践的启示。 非形式化与形式化数学能力之间的差距表明，形式化仍是一项重大瓶颈。然而，近期各模型版本迅速提升的表现说明，我们或许正接近一个临界点：人工智能系统将成为数学工作者的实用工具。若能在形式化验证方面取得成功，便可快速探索证明策略并即时获得正确性反馈，让小众的证明助手专家群体之外的人也能使用形式化方法，并加速研究级数学的形式化。随着模型在 FormalProofBench 等基准测试上的表现持续提升，形式化验证或将从一种专门技术转变为数学研究工具包中的标准组成部分。

未来方向。 若干扩展将加深我们对模型能力的理解 :

1. **探索其他工具配置：** 与 lean_run_code 工具相比，lean_loogle 工具对模型的帮助明显较小；我们还观察到，有相当数量的尝试中，模型未能有效使用 Loogle 工具，反而过度依赖它。采用其他配置，包括语义搜索、搜索本地代码库的能力以及策略级工具，或许能揭示当前搜索界面是否构成瓶颈，还是更深层的推理局限依然存在。2. **消融非形式化问题陈述：** 目前，我们向模型同时提供自然语言陈述和形式化陈述。分析在能否查看非形式化证明的情况下，模型表现有何变化，将有助于明确模型的主要困难究竟在于证明生成，还是在于将数学思想转换为 Lean 语法。3. **扩大多样性与覆盖范围：** 我们的基准测试包含 200 个问题。扩大覆盖范围，纳入更多数学领域并增加问题数量，将提高统计效力，并支持领域专项分析。4. **评估研究级形式化定理证明：** 尽管 FormalProofBench 涵盖高年级本科和研究生水平的数学，但受限于 Mathlib 当前的内容，我们能够合理形式化的问题也受到限制。随着更多数学内容被形式化并纳入 Mathlib，未来版本的这一基准测试可以面向研究级定理，从而为评估人工智能系统解决数学知识前沿问题的能力提供途径。

参考文献

Tudor Achim、Alex Best、Alberto Bietti、Kevin Der、Mathïs Fédérico、Sergei Gukov、Daniel Halpern-Leistner、Kirsten Henningsgard、Yury Kudryashov、Alexander Meiburg、Martin Michelsen、Riley Patterson、Eric Rodriguez、Laura Scharff、Vikram Shanker、Vladmir Sicca、Hari Sowrirajan、Aidan Swope、Matyas Tamas、Vlad Tenev、Jonathan Thomm、Harold Williams 和 Lawrence Wu。Aristotle：国际数学奥林匹克级自动定理证明，2025年。网址：https://arxiv.org/abs/2510.01346。

Pawan Sasanka Ammanamanchi 和 Siddharth Bhat。形式化基准测试中的缺陷。发表于*NeurIPS 2025*第五届数学推理与人工智能研讨会，2025年。网址：https://openreview.net/forum?id=gJ2CpndJmI。

Zhangir Azerbayev、Bartosz Piotrowski、Hailey Schoelkopf、Edward W. Ayers、Dragomir Radev 和 Jeremy Avigad。《ProofNet：本科数学的自动形式化与形式化证明》，2023年。网址：https://arxiv.org/abs/2302.12433。

Tamay Besiroglu、Elliot Glazer 和 Caroline Falkman Olsson。《FrontierMath：用于评估人工智能高级数学推理能力的基准测试》，2024年。网址：https://epoch.ai/ frontiermath/tiers-1-4/the-benchmark/。访问日期：2026-03-10。

Kevin Buzzard 和 Richard Taylor。《费马大定理的持续进行中的 Lean 形式化工作》。2025年。访问日期：2026-03-11。

Karl Cobbe、Vineet Kosaraju、Mohammad Bavarian、Mark Chen、Heewoo Jun、Lukasz Kaiser、MatthiasPlappert、JerryTworek、Jacob Hilton、Reiichiro Nakano、Christopher Hesse 和 John Schulman。训练验证器解决数学文字题，2021 年。网址：https://arxiv. org/abs/2110.14168。

Johan Commelin 和 Adam Topaz。纯数学中的抽象边界与规范驱动开发，2023 年。网址：https://arxiv.org/abs/2309.14870。

Leonardo de Moura 等人。Lean 4 定理证明器与编程语言。Lean Project，2021年。

Amir Dembo。概率论：STAT310/MATH230 课程讲义，2024年。网址：https: // web.stanford.edu/~montanar/TEACHING/Stat310A/lnotes.pdf。斯坦福大学。

Zhitao He、Zongwei Lyu、Dazhong Chen、Dadi Guo 和 Yi R. Fung。MATP-Bench：多模态大语言模型能否成为优秀的多模态问题自动定理证明器？，2025年。网址：https://arxiv. org/abs/2506.06034。

Dan Hendrycks、Collin Burns、Saurav Kadavath、Akul Arora、Steven Basart、Eric Tang、Dawn Song 和 Jacob Steinhardt。使用 MATH 数据集衡量数学问题求解能力。*NeurIPS*，2021年。

Ryan Hsiang、William Adkisson、Robert Joseph George 和 Anima Anandkumar。LeanDojo-v2：用于 Lean 中 AI 辅助定理证明的综合性库。载于*NeurIPS 2025* 第五届数学推理与人工智能研讨会，2025年。网址：https://openreview.net/ forum?id=tnx1VvrcAn。

Thomas Hubert、Rishi Mehta、Laurent Sartran、Miklós Z. Horváth、Goran Žuži´c、Eric Wieser、Aja Huang、Julian Schrittwieser、Yannick Schroecker、Hussain Masoom、Ottavia Bertolli、Tom Za-havy、Amol Mandhane、Jessica Yung、Iuliya Beloshapka、Borja Ibarz、Vivek Veeriah、Lei Yu、Oliver Nash、Paul Lezeau、Salvatore Mercuri、Calle Sönne、Bhavik Mehta、Alex Davies、Daniel Zheng、Fabian Pedregosa、Yin Li、Ingrid von Glehn、Mark Rowland、Samuel Albanie、Ameya Velingker、Simon Schmitt、Edward Lockhart、Henryk Michalewski、Nicolas Sonnerat、Demis Hassabis、Pushmeet Kohli 和 David Silver。通过强化学习进行奥林匹克竞赛级形式化数学推理。《自然》，2025年。网址：https://www.nature.com/articles/ s41586-025-09833-y。

Austin Letson、Leopoldo Sarra、Auguste Poiroux、Oliver Dressler、Paul Lezeau、Dhyan Aranha、Frederick Pu、Aaron Hill、Miguel Corredera Hidalgo、Julian Berman、George Tsoukalas 和 Lenny Taelman。《SorryDB：人工智能证明器能否完成现实世界中的 Lean 定理？》，2026年。网址：https://arxiv.org/abs/2603.02668。

Yong Lin、Shange Tang、Bohan Lyu、Jiayun Wu、Hongzhou Lin、Kaiyu Yang、Jia Li、Mengzhou Xia、Danqi Chen、Sanjeev Arora 和 Chi Jin。《Goedel-prover：面向开源自动定理证明的前沿模型》，2025年。网址：https://arxiv.org/abs/2502.07640。

Junqi Liu、Xiaohan Lin、Jonas Bayer、Yael Dillies、Weijie Jiang、Xiaodan Liang、Roman Soletskyi、Haiming Wang、Yunzhou Xie、Beibei Xiong、Zhengfeng Yang、Jujian Zhang、Lihong Zhi、Jia Li 和 Zhengying Liu。《CombiBench：组合数学大语言模型能力基准测试》，2025年。网址：https://arxiv.org/abs/2505.03171。

mathlib 社区。Lean 数学库。载于第九届 *ACM SIGPLAN* 国际认证程序与证明会议论文集，CPP 2020，第 367–381 页，美国纽约州纽约市，2020 年。计算机协会。ISBN 9781450370974。doi: 10.1145/ 3372885.3373824。URL https://doi.org/10.1145/3372885.3373824。

George Tsoukalas、Jasper Lee、John Jennings、Jimmy Xin、Michelle Ding、Michael Jennings、Ami- tayush Thakur 和 Swarat Chaudhuri。PutnamBench：在普特南数学竞赛中评估神经定理证明器，2024年。URL https://arxiv.org/abs/2407.11214。

R. Vakil。《上升之海：代数几何基础》。普林斯顿大学出版社，2025年。ISBN 9780691268668。

Yuhuai Wu、Albert Q. Jiang、WendaLi、Markus N.Rabe、Charles Staats、Mateja Jamnik 和 Christian Szegedy。使用大语言模型进行自动形式化，2022。网址：https: //arxiv.org/abs/2205.12615。

Huaiyuan Ying、Zijian Wu、Yihan Geng、Zheng Yuan、Dahua Lin 和 Kai Chen。Lean Workbook：一个从自然语言数学问题形式化而来的大规模 Lean 题集，2025年。网址：https://arxiv.org/abs/2406.03847。

Zhouliang Yu、Ruotian Peng、Keyi Ding、Yizhe Li、Zhongyuan Peng、Minghao Liu、Yifan Zhang、Zheng Yuan、Huajian Xin、Wenhao Huang、Yandong Wen、Ge Zhang 和 Weiyang Liu。《Formalmath：大语言模型形式化数学推理基准测试》，2025年。网址：https://arxiv.org/abs/2505.02735。

Haoyu Zhao、Yihan Geng、Shange Tang、Yong Lin、Bohan Lyu、Hongzhou Lin、Chi Jin 和 Sanjeev Arora。《Ineq-comp：不等式自动定理证明中符合人类直觉的组合推理基准测试》，2025年。网址：https://arxiv.org/abs/2505.12680。

Kunhao Zheng、Jesse Michael Han 和 Stanislas Polu。《MiniF2F：形式化奥林匹克级数学的跨系统基准》，2022年。网址：https://arxiv.org/abs/2109.00110。

A 附录

A.1 示例问题与模型回答

我们展示基准测试中的示例问题，以及两个模型给出的回答，以说明失败与成功的证明尝试之间的差异。

问题 1. 证明：一个 A-概形是分离的（相对于 A），当且仅当它在 $\mathbb{Z}$ 上是分离的。此问题选自 Vakil 的《渐升之海》（Vakil，2025年）（第11.2章，练习E）。

```
导入 Mathlib打开 代数几何变量 {A : 类型*} {X : 概形} [交换环 A](结构_X : X --> 谱 (
交换环范畴.of A))定理 代数几何_vakil_ch_11_2_E : 分离 结构_X <-> 分离
(instOverTerminalScheme X).hom := by抱歉
```

代码清单2：Vakil《渐升之海》第11.2章中分离性问题的 Lean 4 形式化。

GPT 5.2（正确）。以下证明成功通过编译。它显式构造了 IsSeparated 实例（通过 IsSeparated.of_comp），并处理了到终对象的态射复合。

```
by
  simpa 使用
    (展示 分离结构_X <-> 分离
        (范畴论.极限.终对象.来自 X) 由此可得
      constructor
        *引入 h
        有 hterm:
            分离
              (范畴论.极限.终对象.来自
                (谱 (交换环范畴.of A))) := 证明
          推断 -实例
        有 hcomp:
            分离
              (范畴论.CategoryStruct.comp struct_X
                (范畴论.Limits.终对象.来自
                  (谱 (交换环范畴.of A)))) :=

    (范畴论.MorphismProperty.复合稳定性.comp_mem
            (P := @代数几何.分离)
            结构体 _X
            (范畴论.极限.终对象.来自
              (Spec (交换环范畴.of A))) h hterm)
        simpa [范畴论.极限.终对象.comp_来自 struct_X]
```

```
        使用 hcomp
    * 引入 h
      有 hcomp :
          分离
            (范畴论.CategoryStruct.comp struct_X
              (范畴论.Limits.终对象.来自
                (谱 (交换环范畴.of A))): = 由
        simpa [范畴论.极限.终对象.comp_from struct_X]
          使用 h
      有I:
          分离
            (范畴论.CategoryStruct.comp struct_X
              (范畴论.Limits.终对象.来自
                (Spec (CommRingCat.of A)))) :=
        hcomp
      exact
        代数几何.IsSeparated.of_comp struct_X
          (范畴论.极限.终对象.来自
            (谱 (交换环范畴.of A)))
```

清单 3：GPT 5.2 的回答（正确）：编译成功。

Claude Opus 4.5 Thinking（不正确）。以下证明尝试的整体结构是正确的，但由于 IsSeparated 的实例解析问题而失败。

```
by
  有 键 : 结构_X ≫ 极限.终对象.来自
      (谱 (交换环范畴.of A)) =
      (instOverTerminalScheme X).hom: = 由
    应用极限.终对象.hom_ext
  constructor
  * 引入 h
    rw [← key]
    精确地 inferInstance
  * 引入 h
    有 h' : 分离 (struct_X ≫
        极限.终对象.来自 (谱 (交换环范畴.of A))) := 由
      rw [key]; exact h
    exact 分离.of_comp struct_X
      (极限.终对象.来自 (谱 (交换环范畴.of A)))
```

Li图 4：Claude Opus 4.5 Thinking 的回答（错误）：因实例解析问题而失败。

这个例子展示了一种常见的失败模式：Claude Opus 4.5 (Thinking) 找到了正确的证明策略，指出这两个态射可以通过与终对象态射复合而建立联系，但在 Lean 的类型类解析环节失败了。GPT 5.2 则通过显式构造所需的实例，并借助复合和化简引理谨慎处理分离类型类，成功完成了证明。

问题 2. 证明：在可测空间 $(\Omega\ \mathcal{F})$ 上，具有“连续性”性质 $B_n \in \mathcal{F}$, $B_n \downarrow \varnothing$, $\mu(B_n) < \infty \implies \mu(B_n) \to 0$ 的有限可加非负集函数 μ 必须是可列可加的，前提是 $\mu(\Omega) < \infty$。本题取自 Amir Dembo 的概率论讲义（Dembo，2024）（练习 1.1.5），并列于列表 1 中。

Claude Opus 4.5 Thinking（正确）。以下证明可成功编译。它通过将 $\bigcup_i A_i$ 分解为有限并与尾项 $B_n = \bigcup_{k \geq n} A_k$，证明 $B_n \downarrow \varnothing$ 和 $\mu(B_n) \to 0$，进而得出结论：NNReal.hasSum_ 当且仅当 _ 趋于 _nat。

```
byhave hEmpty : 测度 empty = 0 := by have h := hAdd empty e
mpty MeasurableSet.empty MeasurableSet.empty disjoint_bot_left
simp only [Set.union_empty] at h have : 测度 empty + 测度 empty =
测度 empty := h.symm have h3 : 测度 empty + 测度 empty - 测度
empty =测度 empty - 测度 empty := by rw [this]simp at h3; exact h3
set B := fun n => iUnion k >= n, A k with hBdef have hB_meas :
forall n, MeasurableSet (B n) :=fun n =>
```

MeasurableSet.biUnion (Set.to_countable _) (fun k _ => hA k) have hB_anti : 反单调 B := by intro m n hmn apply Set.biUnion_subset_biUnion_left intro k hk; exact le_trans hmn hk have hB_inter : iInter n, B n = empty := by rw [Set.eq_empty_iff_forall_not_mem]intro x hxsimp only [Set.mem_iInter] at hx have h0 := hx 0; rw [hBdef] at h0 simp only [Set.mem_iUnion, ge_iff_le] at h0 obtain <k, -, hxk> := h0 have hk1 := hx (k + 1); rw [hBdef] at hk1 simp only [Set.mem_iUnion, ge_iff_le] at hk1 obtain < j, hjk, hxj> := hk1 have hkj : k /= j := by omega exact Set.disjoint_iff.mp (hDisj hkj) <hxk, hxj> have hBtends : 趋于 (fun n => 测度 (B n)) 趋于正无穷 (𝒩 0) :=hCont B hB meas hB anti hB inter¯ ¯ ¯have hUnion : forall n, (iUnion i, A i) =(iUnion k < n, A k) \cup B n := by intro n; ext x simp only [Set.mem_union, Set.mem_iUnion, exists_prop, hBdef] constructor * rintro <k, hxk>by_cases hkn : k < n * left; exact <k, hkn, hxk> * right; exact <k, not_lt.mp hkn, hxk>* rintro (<k, -, hxk> | <k, -, hxk>) <;> exact <k, hxk>have hDisjParts : forall n, 不相交 (iUnion k < n, A k) (B n) := by intro n; rw [Set.disjoint_iff]intro x <hx1, hx2>simp only [Set.mem_iUnion, exists_prop, hBdef] at hx1 hx2 obtain <k1, hk1n, hxk1> := hx1 obtain <k2, hk2n, hxk2> := hx2 have hne : k1 /= k2 := by omega exact Set.disjoint_iff.mp (hDisj hne) <hxk1, hxk2>have hFinMeas : forall n, MeasurableSet (iUnion k < n, A k) := fun n =>MeasurableSet.biUnion (Set.to_countable _) (fun k _ => hA k) have hFinAdd : forall n, 测度 (iUnion k < n, A k) =sum k in Finset.range n, 测度 (A k) := by intro ninduction n with| zero => simp [hEmpty] succ n ih =>rw [Finset.sum_range_succ, ← ih]have hunion : (iUnion k < n + 1, A k) =(iUnion k < n, A k) \cup A n := by ext x; simp only [Set.mem_iUnion, Set.mem_union, exists_prop] constructor *rintro <k, hkn, hxk>by_cases hkn' : k < n * left; exact <k, hkn', hxk>* have heq : k =n :=by omega right; rw [heq] at hxk; exact hxk *rintro (<k, hkn, hxk> | hxn) * exact <k, Nat.lt_succ_of_lt hkn, hxk>* exact <n, Nat.lt_succ_self n, hxn>have hdisj : 不相交 (iUnion k < n, A k) (A n) := by rw [Set.disjoint_iff]; intro x <hx1, hx2>simp only [Set.mem_iUnion, exists_prop] at hx1 obtain <k, hkn, hxk> := hx1 have hne : k /= n := by omega exact Set.disjoint_iff.mp (hDisj hne) <hxk, hx2>rw [hunion]; exact hAdd _ _ (hFinMeas n) (hA n) hdisj have hKey : forall n, 测度 (iUnion i, A i) =(sum k in Finset.range n, 测度 (A k)) + 测度 (B n) := by intro ncalc 测度 (iUnion i, A i)= 测度 ((iUnion k < n, A k) \cup B n) := by

```
            改写 [hUnion n]
      _ = 测度 (iUnion k < n , A k) + 测度 (B n) :=
            hAdd _ _ (hFinMeas n) (hB_meas n) (hDisjParts n)
      _ = (sum k 属于 Finset.range n, 测度 (A k)) + 测度 (B n) :=
            由 rw [hFinAdd n]
  有 hPartialSums : 趋于
        (函数 n => 对 k 求和 属于 Finset.range n, 测度 (A k))
        趋于正无穷 (𝓝 (测度 (iUnion i, A i))) := 作者
    有 h1: 对所有 n ,
          (对 k 求和 属于 Finset.range n, mu (A k)) =
          mu (iUnion i, A i) - 测度 (B n) := 由
        引入 n
        有 := hKey n
        假设 h2 : 测度 (iUnion i , A i) - 测度 (B n) =
              求和 k 属于 有限集.range n, mu (A k) +
              测度 (B n) - 测度 (B n) := 通过 rw [←此]
        简化 于 h2; 精确应用 h2.symm
    简化_重写 [h1]
    有 : 趋于
          (函数 n => 测度 (iUnion i , A i) - 测度 (B n))
          趋于正无穷 (𝓝 (测度 (iUnion i, A i) - 0)) :=
        趋于.sub 趋于 _常量 _nhds hBtends
    简化 于此; 精确 此
  有 hHasSum : HasSum (函数 n => 测度 (A n))
        (测度 (iUnion i, A i)) := 作者
    rw [非负实数.hasSum_iff_趋于_nat]; exact hPartialSums
  exact hHasSum.tsum_eq.symm
```

列表 5：Claude Opus 4.5 Thinking 的回答（正确）：成功编译。

Gemini 3 Pro（不正确）。以下证明尝试采用了类似的分解策略，但由于度量空间部分的收敛论证存在问题而失败。

```
by
  令 B := 函数 n => iUnion k , A (n + k)

  有 hB_可测: 对于 n, 可测集 (B n) := 证明
    引入 n; 应用 MeasurableSet.iUnion; 引入 k;
    精确地, hA (n +k)

  有hB_反单调 B: = 由
    引入 m n hmn x hx
    简化 [B]在 hx ⊢
    对 hx 使用分情况 <k , hk>
    使用 (n - m) +k
    有 :n +k =m +((n - m) +k): = 由
      rw [add_assoc , Nat.add_sub_of_le hmn]
    rw [这个]在 hk
    exact hk

  具有 hB_交集 : (iInter n, B n) = 空: =通过
    rw [Set.eq_空集_当且仅当_对所有_不属于]
    intro x hx
    simp于 hx
    rcases hx 0 with <k0, hk0>
    rcases hx(k0 + 1) 其中 <k1 , hk1>

    有 neq : k0/= k0 + 1 + k1 := 证明 linarith
    exact (hDisj neq) .ne_of_mem hk0 hk1 rfl

  有 h_极限 : 趋于 (函数 n => 测度 (B n)) 趋于正无穷 (𝓝 0) :=
    hCont B hB _meas hB _anti hB _交

  令S: = 对n => iUnion k 属于Finset.range n, A k
  有hS _可测: 对于所有n , MeasurableSet (S n): =
    函数 n => 可测集双重并
      (Finset.range n).countable_toSet (fun k _ => hA k)

  有 h_分解: 对所有 n, (iUnion k, A k) =S n \cup B n: =通过
    引入 n; 外延 x; 化简 [S, B]
    constructor
```

```
*rintro <k, hk>by_cases h : k < n * left; use k, h * right; use k -
n; rw [Nat.add_sub_of_le (le_of_not_lt h)]; exact hk *rintro (<k,
hk_lt, hk> | <k, hk>) * use k, hk * use n + k, hk have h_disj_SB :
forall n, 不相交 (S n) (B n) := by intro n; rw [Set.disjoint_left]intro
x hx S hx B¯ ¯simp [S, B] at hx_S hx_B rcases hx_S with <k1,
hk1_lt, hk1>rcases hx_B with <k2, hk2>have : k1 /= n + k2 := by
linarith exact (hDisj this).ne_of_mem hk1 hk2 rfl have h_hAdd_n :
forall n, 测度 (iUnion k, A k) = 测度 (S n) + 测度 (B n) := by intro
n; rw [h_decomp n]apply hAdd (S n) (B n) (hS_meas n) (hB_meas
n) (h_disj_SB n) have h_sum_S : forall n, 测度 (S n) =sum k in
Finset.range n, 测度 (A k) :=by intro ninduction n with|zero =>si
mp [S]have : 测度 empty = 0 := by have := hAdd empty empty 可
测集.empty 可测集.empty (disjoint_empty empty) simp at this; exact
this exact this| succ n ih =>rw [Finset.sum_range_succ, ←ih]
have hS_disj : 不相交 (S n) (A n) := by rw [Set.disjoint_left]intro x
hx S hx A¯ ¯simp [S]at hx_S rcases hx_S with <k, hk_lt, hk>
exact (hDisj (ne_of_lt hk_lt)).ne_of_mem hk hx A rfl¯have hS_
union : S (n + 1) = S n \cup A n := by ext x; simp [S];
constructor *rintro <k, hk_lt, hk>rw [Nat.lt_succ_iff_lt_or_eq]
at hk_lt cases hk lt¯*left; use k; assumption; assumption *right;
subst k; assumption *rintro (<k, hk_lt, hk> | hxn) *use k;
constructor; apply Nat.lt_succ_of_lt hk_lt; assumption *use n;
constructor; apply Nat.lt_succ_self; assumption rw [hS_union]rw [
hAdd (S n) (A n) (hS_meas n) (hA n) hS_disj]have h_conv : 趋于 (fun
n => 测度 (S n)) 趋于正无穷 (𝒩 (测度 (iUnion k, A k))) := by apply
Metric.tendsto_nhds.mpr intro eps heps rw [Metric.tendsto_nhds]
at h_lim specialize h_lim eps heps rw [Filter.mem_atTop_sets] at
h_lim rcases h_lim with <N, hN>use N; intro n hn; specialize hN n
hn
```

```
    rw [dist_eq, 实数.dist_eq, abs_eq_self.mpr]
    * rw [h_add_n n]; rw [dist_comm]
      rw [dist_eq_norm]; simp
      rw [← 非负实数.coe_add]
      rw [add_sub_cancel_left]; simp
      rw [dist_eq_norm] 在hN; 简化 在hN; 精确 hN
    * simp

  对称性；应用 HasSum.tsum_eq
  rw [非负实数.hasSum_iff_趋于_nat]
  simp_rw [←h_sum_S]; exact h_conv
```

清单 6：Gemini 3 Pro 的回答（不正确）：在收敛步骤中失败。

两个证明遵循相同的策略，将 $\mu(\bigcup_i A_i)$ 分解为有限部分和与一个趋于零的尾项，但处理收敛步骤的方式不同。Claude Opus 4.5 (Thinking) 全程使用 NNReal API，而 Gemini3 Pro 尝试对 NNReal 表达式使用实值度量引理（Metric.tendsto_nhds、dist_eq），导致 Lean elaborator 无法解决的类型不匹配。

A.2 工具指导提示

```
工具_指导_模板 = """预算：总共 {max_turns} 轮。
这是你提交以下内容所拥有的总轮数：
定理的最终 Lean 4 证明。每一轮最多允许
最多调用 6 次工具（超过 6 次的调用将被静默丢弃）。

关键评分规则：除非你调用
submit_proof，否则得分为 0。未提交将自动判定失败。

工具：-
  lean_run_代码：测试证明，并根据错误反复迭代
- lean_loogle：搜索现有引理和定义
  来自 Mathlib（请谨慎且高效地使用此工具）
- submit_proof：提交最终证明——这是获得任何分数的必要条件

重要——工具调用策略：
- 每轮最多可进行 6 次工具调用。请高效使用。
- 示例：一轮中，你可以调用 lean_loogle 两次，并且
  lean_run_code 4 次（共 6 次）。
- 或者调用 lean_run_code 6 次，同时测试多个证明
  变体。
- 在最后一轮中，务必在工具调用中包含 submit_proof！

请明智地使用这些工具；以下是一些建议：
- 即使不完美，也要提交你最好的尝试——不完美的
  提交也胜过不提交
- 如果剩下的轮次不多了，请停止探索
  并尽快调用 submit_proof
- 如果你很早就确信自己的证明正确，请调用
  立即调用 submit_proof
- 不要在未调用 submit_proof 的情况下耗尽全部 {max_turns} 轮
  submit_proof
"""
```

图 6：附加到用户提示中的工具指导模板。