arXiv:2611.29811v1 [math.HO] 22 Nov 2026

用于数学研究的生成式人工智能札记

Shurui Liu
srliu@stanford.edu

斯坦福大学数学系

2026年11月

摘要

本文介绍生成式人工智能如今如何用于解决和检验研究级数学问题，重点关注智能体及其工具框架。对于证明智能体 Rethlas、Danus 和 Pharos，以及验证智能体 OSVerify，我们阐释其设计原则，将它们分别与各自的基础模型进行比较，展示案例研究，并说明如何使用和参与改进这些工具。我们还简要回顾数学人工智能的发展历史及学界对此的反应。

*2020*年数学主题分类。主要分类 68T42；次要分类 68T20、68T50、68V15、00A35、01A61。

目录

1 引言 **2**
1.1 概述 .2 1.2 哲学观点 .3 1.3 论文结构 .5 1.4 人类使用与人工智能使用声明 .5 1.5 致谢 .5

2 数学人工智能简史 **5**
2.1 旧时代：计算机辅助证明 .6 2.2 不那么久远的时代：机器学习方法 .6 2.3 语言模型与智能体时代 .7 2.4 对数学人工智能的看法 .9

3 什么是数学研究 **10**
3.1 研究循环 .10 3.2 为什么研究不同于竞赛 .11

4 什么是工具框架和智能体 **12**
4.1 什么是大语言模型 .12 4.2 第一个定义：智能体 =模型 + 工具框架 .13 4.3 第二个定义：工具框架 =上下文管理 .14 4.4 通用型工具框架（Codex、Claude Code）及其局限性 14

4.5 为什么要研究工具框架和智能体 .154.6 设计原则：如何设计工具框架和智能体 .16

5 用于证明和问题求解的智能体：Rethlas、Danus、Pharos 165.1 Rethlas：生成–验证循环 .17

5.1.1 与基础模型的比较 .175.1.2 智能体工作流设计 .175.1.3 Rethlas 的设计原则 .185.1.4 案例研究：版权与理解 .195.1.5 为什么生成–验证循环还不够 .20

5.2 Danus：智能体研究组.21
5.2.1 设计.215.2.2 案例研究：拟阵构造集的切丛类.235.2.3 局限性.24

5.3 Pharos：面向更强模型的 Danus.245.3.1 设计.245.3.2 案例研究 1：Hartshorne 猜想的代数证明. . . . . 265.3.3 案例研究 2：创建一个统一两个问题的框架. 26

6 用于验证的智能体 验证：OSVerify 27
6.1 动机与目标.276.2 设计：提出—验证循环 .276.3 与 LLM 评审和多数投票的比较 .286.4 与前沿模型的比较：数学研究中的案例研究 . . . . . 296.5 OSVerify 作为元算法 .30

7 数学家如何使用并作出贡献 30
7.1 选择工具：别用大锤砸核桃 .307.2 通过编程智能体安装并运行智能体 .317.3 手动运行和自定义智能体.317.4 通过操作代理使用 Pharos .337.5 良好实践 .337.6 如何贡献 .34

8 其他领域中的未来方向和应用 348.1 开放方向 .34

A 术语表 **35**

1 引言

1.1 概述

1.1.1. 动机。 这些笔记总结了我在2026年对用于数学研究的人工智能的探索。内容拓展自我在斯坦福大学、昆山杜克大学和 Cursor 所作的演讲，听众包括数学家和计算机科学家；此外，也融入了我在斯坦福大学讲授 Math 289 课程（人工智能与数学）时的经验。这些笔记旨在

记录我的理解，并帮助弥合数学家与人工智能研究人员之间的信息鸿沟。

1.1.2. 用于数学研究的自然语言智能体。 我们关注数学人工智能的非形式方面，而不是形式方面。也就是说，这里的问题求解、证明生成和验证，指的是用自然语言（如英语）进行推理，而非使用形式语言（如 Lean 4）。

1.1.3. 偏见与局限。 这些笔记中的大部分内容和观点，都源自我截至2026年11月在数学和数学人工智能领域的个人研究。因此，这些内容带有偏见，仅代表我个人的观点。它们并非旨在代表数学界，也无意涵盖所有相关工作或方法。

1.1.4. 本文讨论的智能体。 表 1 概述了第 5 节和6节讨论的系统。它们都是开源的。第 5 节介绍能够求解问题和证明定理的系统：Rethlas、Danus 和 Pharos。第 6 节介绍 OSVerify，这是一种用于检查证明的验证智能体。对于每个系统，我们首先将其能力与基础模型进行比较，然后介绍其设计，接着讨论案例研究，最后说明其局限性。

系统	Idea	它所模拟的对象	参考对象
Rethlas	一个生成器和一个验证器循环协作	一名数学家和一名审稿人	[Ju+26]
Danus	一个共享以下内容的智能体研究组 已验证事实的有向无环图的智能体研究组	初级学生研究组 配有一名管理员和一名外部专家顾问	[Liu+26]
Pharos	为更强大的模型重新设计的 Danus：路由板、数值工具、深度思考等。	一个由顶尖博士组成的研究团队，配备一位 PI	[Don+26]
OSVerify	一个智能体提出错误，另一个对其进行验证	一位严谨而严格的审稿人	[Nie+26]

表1：第5 节和6节讨论的系统。

1.1.5. 精选比较。 表2 汇总了将基础模型与我们的系统进行比较的代表性案例研究。更多详情见 §5 和 §6。

1.2 哲学观点

1.2.1. 一些总体看法。 数学界对于在研究中使用人工智能持有各种不同观点。我猜大多数人都会认同以下几点。

1. 思想与观点的多样性有益于数学和人类。2. 人工智能已经非常擅长解决问题和生成证明，不过它的能力因问题和领域而异。但数学研究不只是问题求解。3. 人工智能还不是能解决所有问题、完成所有任务的魔法盒子，尽管人们对未来有着不同的预测。

问题	域	仅使用模型	使用我们的系统
余特征 [朱+26]	代数的群	GPT-5.5 Pro：错误证明	Rethlas (GPT-5.5)：正确证明，44 分钟
拟阵KL多项式 [CL26]	组合学	GPT-5.6 Sol Ultra：不完整	Rethlas（GPT-5.6 Sol）：反例，5 小时
拟阵切丛类 [CLG26]	组合学	GPT-5.5 Pro：否解	Danus（GPT-5.5）：已解决，~5 天
Hartshorne 猜想，再次得到证明 [Don+26]	代数几何	GPT-6-Astra 与专家指导：否解	Pharos（GPT-6 Astra）：通过喷射获得的新证明，20 小时，自主
Manin 常数，评审 [CLN26]	数论	Claude Opus 5.5, Claude Fable 5.1, Codex（GPT-6 Astra）：12 轮，未发现错误 已发现	OSVerify（GPT-5.6 Sol）：发现一处错误，已建议修复 已建议
11篇OpenAI预印本 [Ope26d]	各种	OpenAI 的内部模型：错误忽略了	OSVerify（GPT-6-Astra）：在每个.....中发现了错误

表2：仅使用基础模型与我们的系统：部分结果。

4. 数学家看重富有智识趣味且具有挑战性的工作；机器可以轻易完成的工作，最终会失去价值。5. 即使人们对此看法不同、行动各异，我们也应始终至少怀有促进数学、科学和人类进步的意愿。6. 在社群中，诚实与声誉始终至关重要。

1.2.2. 一些个人看法。 我个人的立场由上述观点以及以下内容构成。

1. 数学的发展与数学家的存续可能会背道而驰。我们当然应以促进前者为目标。但数学家，尤其是那些不愿使用人工智能、选择依靠自身创造力的人，应当受到尊重和保护，因为没人能确定长远来看是否仍需要他们。2. 人类，或者至少部分人类，仍能在人工智能之上创造价值。要让这一点继续成立，我们必须以负责任的态度对待年轻一代；他们受到教育、社群评估体系、社交媒体以及杰出数学家的塑造。3. 在人工智能辅助证明中，有三类人会为人类对数学的理解作出贡献：（i）构建人工智能工具的人，（ii）借助工具解决问题的人，以及（iii）理解解并帮助他人理解它的人。我认为他们的贡献可比较为（i）$\ll$（ii）$<$（iii}。

1.2.3. 建议。 这些笔记侧重于技术实践，而非哲学辩论。不过，我还是想提出几点温和的建议。

- 请不要使用人工智能系统抢先窃取他人的成果。

- 请不要在自己不懂的领域大量生成人工智能垃圾内容。
- 请不要去解决某个著名猜想的一个薄弱或无趣的变体，然后声称自己在该猜想本身上取得了重大进展。

1.2.4. 其他观点。 我们在第 2.4 节简要概述数学界的反应和舆论。

1.3 论文结构

第 2 节简要回顾用于数学研究的人工智能的发展历程，以及数学界的反应。第 3 节探讨数学研究的内涵，以及我们为何认为它不同于竞赛数学和国际象棋。第 4 节回顾大语言模型、工具框架和智能体的基本概念，并讨论数学智能体的设计动机和原则。第 5 节和 6 节介绍我们用于问题求解和验证的智能体：其设计、与基础模型的比较，以及案例研究。第 7 节说明如何使用这些系统并为其作出贡献，第 8 节则指出一些未来方向。

1.4 人类使用与人工智能使用声明

本文主要由作者撰写。表格和图示是在作者指导下借助人工智能工具生成，随后由作者修订。生成式人工智能工具还用于改进阐述（更正拼写和语法错误、改写句子）、发现不准确之处，以及协助查找相关参考文献。本文呈现的所有哲学观点和概念性理解，均由作者通过独立思考并与他人讨论而形成；部分计算机科学技术背景知识则是在人工智能工具的帮助下学习的。作者对内容的准确性承担全部责任。

1.5 致谢

本笔记扩展自数学人工智能研讨会、斯坦福大学数学系的 Math 289、斯坦福大学计算机科学系的 Graphics Cafe 研讨会、昆山杜克大学的数学人工智能研讨会以及 Cursor 上的演讲。S. Liu 感谢各位组织者和主办方，以及听众们所带来的启发性讨论，尤其感谢 Otis Chodosh、Andrea Montanari、Ravi Vakil、Lexing Ying、George Nakayama、Zander Majercik、Jiajun Ma、Xiaoqian Xu 和 Shengtong Zhang。

S. Liu 感谢各位合作者在实践中对他的教导，尤其感谢 Bin Dong、Guoxiong Gao、Haocheng Ju、James Zou、Haotian Ye、Fan Nie、Bin Wu、Zeming Sun 和 Jiedong Jiang。他还感谢 Brian Conrad、Fred Rajasekaran、Henry Bosch、Ronnie Cheng、Haosen Wu、Rosie Shen、Vaughan McDonald、Richard Taylor、Yanan Jiang、Travor Liu、Qiyue Tang、Zhuo Zhang、Carl Schildkraut、Michael Ren、Zeyu Wang、Romain Speciel、Judson Kuhrman、Selim Amar、Yizhen Chen、Hongrui Chen 和 Alec Shelley 所进行的富有意义的讨论，并感谢他的导师 Xinwen Zhu 一直以来的支持。

2 数学人工智能简史

本节并非旨在全面综述人工智能辅助成果。我们选取了一些示例，并讨论这些示例所反映的情况，以及我们对这一快速进展的个人看法。

2.1 旧时代：计算机辅助证明

早在大语言模型和智能体出现之前，就已经有许多计算机辅助证明。

2.1.1. 示例。

等式逻辑中的穷尽符号搜索 1996年，McCune 的证明器 EQP 证明了每个 Robbins代数都是布尔代数，这个问题自 20 世纪 30 年代以来一直悬而未决 [McC97]。

穷尽数量庞大的有限情形 Appel 和 Haken 对四色定理的证明（1976 年） [AH77] 引发了一场旷日持久的争论：一项无人类能够全面审查的证明，究竟算不算证明。不过，计算机依赖的是人类设计的证明策略，该策略将原问题归约为有限多个情形。

形式语言与交互式定理证明器 Lean [MKADR15]是一种交互式定理证明器：定义、陈述和证明都用形式语言编写，并由一个小型可信内核机械地检查。由社区维护的库 Mathlib [该库20]提供了一个庞大且统一的形式化数学体系，涵盖了本科课程的大部分内容以及重大研究生层次材料。Scholze 提出的 Liquid Tensor Experiment 挑战 [Sch22] 由 Commelin 领导的团队完成，目标是用 Lean 形式化 Clausen 和 Scholze 关于液体向量空间的一项困难定理。

2.1.2. 蛮力与确定性程序。 上述所有示例，或多或少都是由确定性程序进行的蛮力搜索。计算机并未提出新想法；它只是比人类计算得更快。原则上，只要有足够的时间，人类也能完成同样的任务。

2.2 不那么久远的时代：机器学习方法

从大约 2016 年起，机器学习以多种角色进入了数学领域。

2.2.1. 指导 *Lean* 证明器。 神经前提选择 [Irv+16]随后，在形式证明上训练的语言模型 [PS20] 学会了建议下一步策略或引理。LeanDojo 等工具[Yan+23] 以及面向 Mathlib 的语义搜索引擎 [GJJQD24]均借助机器学习构建。随后出现了一系列能力不断增强的形式证明器（DeepSeek-Prover、Kimina、Seed-Prover[Che+25]、Aristotle [Ach+25]），定理陈述自动形式化也随之发展 [Gao+24；Wan+25]。

2.2.2. 示例的数值发现。 Davies 等人 [Dav+21]对经过训练的网络使用归因方法，引导数学家在纽结理论中发现了一个新定理，并推动了关于 Kazhdan–Lusztig 多项式的组合不变性猜想的研究。Wagner 使用强化学习在图论中找到了反例 [Wag21]。PatternBoost [CEWW24]和 AlphaEvolve [Nov+25] 将这一思想扩展到了组合学和分析学中的许多构造。在这些系统中，模型提出方案，程序进行评分；当目标是一个可计算的数值时，它们的表现最佳。

2.2.3. 用于国际数学奥林匹克几何题的符号演绎。 AlphaGeometry [TWLHL24] 将语言模型与符号演绎引擎相结合，解决了 30 道国际数学奥林匹克几何题中的 25 道。2024年7月，AlphaProof 和 AlphaGeometry 2 在国际数学奥林匹克中达到银牌水平，其证明采用 Lean [Hub+25]。

2.2.4. 智能与蛮力之间的界限。这些方法超越了前一时代的确定性程序：决定接下来尝试哪种策略、考察哪个示例或添加哪种构造的，不再是固定规则，而是经过训练的模型。这是一种直觉，使蛮力无法完成的搜索成为可能。然而，上述示例仍具有蛮力搜索的特点。搜索局限于预先设定的狭小空间；每个候选项都由 Lean内核、评分程序或符号引擎自动检查。模型的作用是剪枝搜索树。尽管聪明的搜索与真正的智能之间的界限究竟在哪里尚有争议，但下一时代的示例在性质上有所不同。

2.3 语言模型与智能体时代

2.3.1. 数学竞赛与唾手可得的*Erdős*问题。

- **2025年7月：国际数学奥林匹克金牌**。OpenAI的一款实验性通用推理模型在国际数学奥林匹克的六道题中解出了五道（35/42，达到金牌水平），并在竞赛时限内撰写了自然语言证明；这些证明由前国际数学奥林匹克奖牌得主评分，而非由国际数学奥林匹克 [Wei25]评定。Google DeepMind的Gemini Deep Think在获得官方认证的情况下取得了相同分数。

- **2026年1月：Erdős问题**。GPT-5.2 Pro 证明了几个 Erdős问题，并由 Harmonic 的 Aristotle 在 Lean 中完成形式化 [Ach+25]；陶的公共维基记录了人工智能对 Erdős问题的贡献 [Tao+26]。这些问题通常被认为是容易解决的问题。其中一些实际上早已在文献中得到解决：例如，厄尔多斯问题第397号最终被发现就是2012年中国国家队选拔考试第一天的第3题。另一些问题虽然当时尚未解决，但难度较低，凭借已知方法即可解决，例如厄尔多斯问题第728号。

2.3.2. 组合学与较小的研究结果。

- **2026年2月：Aletheia**。Google DeepMind 的 Aletheia 是一个基于 Gemini Deep Think 构建的研究智能体，能够以自然语言生成、验证和修订证明；它在没有人类干预的情况下，就算术几何中的特征权发表了一篇论文 [Fen26], 参与了与数学家合著的多篇论文，并在一项对700个 Erdős问题进行研究的工作中自主解决了其中四个尚未解决的问题 [Fen+26b]。

- **2026年2月：First Proof**。11 位数学家发布了 10 条来自当前研究的引理 [Abo+26]。经专家审查后，OpenAI内部模型生成的解答中约有 5 条被判定为正确 [Ope26b], Aletheia 的解答中有 6 条获得了多数评审者认可 [Fen+26a]。

- **2026年8月：668 阶 Hadamard矩阵**。 8月12日，L. Alpöge、P. Voinov 和 S. Reynolds-Haertle 在 Claude 的协助下，宣布构造出了此前尚无已知结果的、低于 2000 阶的全部十二种 Hadamard 矩阵 [AVR26]。其中包括 668 阶，这是自 2005 年构造出 428 阶矩阵以来，Hadamard猜想中最小的未解决情形。随后，Egan、Kotsireas、London 和 Ó Catháin 对这些矩阵的结构进行了分析[EKLÓ26]。

2.3.3. 可发表且重要的结果。 一些在线列表记录了数学研究中的重大人工智能辅助成果，例如 [Mat26]。此处仅列出部分受到公众关注的成果。

- **2026年5月：单位距离。** OpenAI 的一个内部模型推翻了埃尔德什猜想，即 n 个平面上的点至多确定 $n^{1+o(1)}$ 条单位距离，其依据是来自代数数论的一种构造（数域与戈洛德–沙法列维奇论证） [Ope26c]。由九位数学家组成的小组验证并阐释了该论证 [Alo+26]。由于该方法具有新颖性，许多数学家认为这一结果足以发表在诸如《数学年刊》这样的顶级期刊上。数日后，Anthropic 报告称，Claude Mythos 已独立找到一个（略弱一些的）否证。

- **2026年7月：雅可比猜想。** L. Alpöge（Anthropic）宣布了一个显式多项式映射 $\mathbb{C}^3 \to \mathbb{C}^3$，其次数为 7，雅可比行列式为常数，且不是单射；该映射借助 Claude Fable 5 找到，从而在每一维上推翻了雅可比猜想 $n \geq 3$。该反例在一天之内于 Isabelle 中得到验证 [RHQ26]，并由 Tao 加以解释[Tao26]。该猜想十分著名，但专家主要关心二维情形，而这一情形仍远未解决。

- **2026年8月：十项进展。** 8月1日，OpenAI 发布了其尚未公开的模型 Astra 在数学和理论计算机科学领域十个开放问题上的结果，并为这十个问题全部提供了 Lean 证明证书 [Ope26e]。这些结果包括证明非 sofic 群的存在，以及推翻 Connes 刚性猜想。不过，该领域的研究者提出了一种可能性：其中的关键思想源自他们未发表的工作。

- **2026年8月：上的复结构。** S^6。8月23日，L. Alpöge 发布了一份长达108页的手稿。该手稿由 Claude 的内部版本在他的指导下生成，构造了一个紧致复三维流形的单参数族，这些流形与 S^6微分同胚。这回答了 Hopf 于1947年提出的问题：六维球面是否存在复结构。P. Engel 给出了一个自成一体的阐述 [Eng26], ，B. Alexeev 则在 Lean 中形式化了该证明。截至本文撰写时，专家认为这一构造具有可信度，审查仍在继续。

- **2026年9月：带力项的纳维–斯托克斯方程。** OpenAI 发布了关于受力的 三维纳维–斯托克斯方程有限时爆破的证明，该证明附有一份约 230,000 行的 Lean 证明证书，由一大群智能体生成 [Ope26a]。这并非无外力的千禧年难题，而且其署名归属以及形式化陈述是否忠实地刻画了非形式化陈述，都存在争议 [BCH26]。

- **2026年10月：722 篇手稿。** OpenAI 发布了 722 篇手稿，涵盖 372 个成果系列。这些手稿由一个内部模型从约 4,000 个尝试解决的问题中生成，署名为“OpenAI”，没有人类作者；发布时约有一半尚未经过验证 [Ope26d]。几天之内，其中三篇因符号错误而被撤回，十四篇进行了修订 [Ret26]。所声称的结果包括 Kakeya 集的完整维数 $\mathbb{R}^4$ 以及拟黎曼猜想。如果正确，其中许多成果将能够在顶级期刊上发表，并会成为各自领域的里程碑。

- **2026年11月：黎曼猜想。** 11月20日，L. Alpöge 宣布，他与 Claude 合作找到了黎曼猜想的证明，并且该方法也适用于广义黎曼猜想 [Alp26；Sci26]。该证明并未以传统论文的形式发布，而是以 Lean 形式化的形式发布，据称约有 1700 万行代码，并附有一份简短的说明。

- **2026年11月：无力项的纳维–斯托克斯方程**。同一天，T. Buckmaster 和 L. Alpöge 发布了一份大量借助人工智能完成的证明，论证无外力三维纳维–斯托克斯方程存在有限时爆破，并配有一个约 60 万行的 Lean 库 [BA26]。截至本文撰写时，该证明尚未经过独立核查，证明中的数值部分也尚未发布。

2.4 关于数学人工智能的看法

2.4.1. 意见分歧的数学界。 数学界的意见分歧十分明显。例如，可以参阅社群博客 *Proofs and Prompts* [26] 以及美国数学学会关于数学与人工智能的网页 [Ame26]。各种反应涵盖了完整的光谱。最极端的一端是彻底反对。例如，2026年8月，M. Weinreich 阐述了彻底反对在数学中使用人工智能的理由 [Wei26]。人类数学协会成立于2026年9月，旨在保护数学作为一项人类事业 [Ass26]。另一极端则是全盘支持在没有人类参与的情况下生成的人工智能数学。介于两者之间，许多人主张以透明且负责任的方式使用人工智能。随着基础模型不断进步，越来越多的重要问题得到解决，这些观点也在持续演变。

2.4.2. 一个来自科幻小说的类比。 刘慈欣的《地球往事》三部曲 [Liu14；Liu15] 的故事始于人类得知三体人——一个先进得多的外星文明——已派出一支舰队，将在大约四个世纪后抵达地球。舰队前方先遣而来的，是被称为“智子”的粒子大小的计算机，它们能够观察人类所说或所写的一切，却无法窥见人类的思想。故事的大部分篇幅都围绕漫长的等待展开：一个社会如何围绕一种尚未到来的、更高级的智能重新组织自身。人们分成不同阵营，对事实看法一致，却对其后果各执己见。复临论者希望外星人取代他们认为无可救药的人类；拯救派崇敬外星人，并希望拯救自己的世界；胜利主义者和失败主义者则争论抵抗能否成功；少数“面壁者”受命在唯一无法被观察的地方——自己的头脑中制定计划。第二卷进一步描绘了宇宙如同一座“黑暗森林”的意象：任何暴露自身的文明都有遭到毁灭的风险。

我们认为，将其与数学类比有三个好处。第一，二者的情形结构相同：一种非人类智能的能力正在增长，但增长时间表尚不确定；而一个社群必须在还不知道这种智能最终会有多强大之前，决定如何应对。第二，值得关注的差异在于人类的各种应对方式，而派别分类法所记录的正是这些差异。第三，智子有一个直接的对应物：数学家发布的任何内容都可能被人工智能系统读取，也可能被其使用，并用于训练。不过，这种类比仅限于反应模式。下文的标签都是漫画式的概括，用来描述各种态度，并不表示赞同或谴责这些态度。我们宽泛地借用了这些名称，也从其他地方取了几个，因为小说中的派别名称已经不够用了。

2.4.3. 反应分类法。

复临论者认为，人工智能已经全面超越了人类数学，因此手工做数学是对时间和资源的浪费。较温和的观点则认为，人类数学本身存在缺陷，或者至少速度太慢 [DAR25]，，而人工智能与人类协同工作将极大加快数学的发展，从而拯救我们。

拯救派 认为，我们应当投入精力改进人工智能，帮助它摆脱训练数据的“凸包”，以及出版物对人工智能生成的作品的“歧视”。

胜利主义者 期待迅速取得胜利 [刘15]。他们认为，人工智能当前的能力只是一个过渡阶段：它迟早会遇到瓶颈，或者永远无法离开训练数据的凸包，又或者业界会失去兴趣并停止投资。在每种情况下，人类都会重新占据领先地位。

投降主义者 承认人类数学仍然具有价值和未来，但认为自己身上毫无价值，也看不到未来，于是得出结论：他们能做的只有拥抱人工智能来谋生。

持久战派 承认人工智能在不断进步，但也观察到它在阐述、理论构建和真正的创新方面仍然薄弱。他们的策略是持续寻找人工智能做得不好的事情，并在这些地方增添人类价值。

面壁者 [刘15] 像小说中与他们同名的人物一样，把自己的想法藏在心里，只在头脑中进行谋划，因为一切说出或写下的内容都会被窃听。对他们而言，数学已经变成了一片黑暗森林：任何公开的想法都可能被他人或人工智能夺走并抢先发表，因此想法必须保密，直到工作完成。

逍遥者， 得名于逍遥 （“《逍遥游》”）庄子 [朱13],只是享受生活，以及理解数学的乐趣（参见 [屠 94]）。他们以文献为模型，其中既有职业作家，也有业余爱好者；而文献的黄金时代，只有在人人都能读写之后才会到来。

蒸汽派认为数学正经历一场工业革命，数学研究进入了一个大规模生产的时代，这与 Tao 所描述的工业规模数学相近 [赢得了24]。

2.4.4. 评估与招聘。 无论持何种立场，数学界的评估与招聘系统都必须改变。一些招聘委员会已经开始发布建议，并说明他们在评估应用方面的变化 [Sev26]。

3 什么是数学研究

本节不涉及技术细节。在第 3.1 节中，我们将数学研究分解为六个阶段；在第 3.2节中，我们将讨论研究级数学为何不同于竞赛。

3.1 研究循环

3.1.1. 六个阶段。 数学研究大致可以分解为由六个阶段组成的循环（图 1）。

1. **发现：** 观察现象，并提出猜想，以统一或概念化的方式解释这些现象。2. **问题求解：** 证明或证伪猜想，或计算答案。3. **验证：** 检查一个解是否正确。4. **理解：** 内化证明为何成立，明确其中关键且新颖的思想，以及它如何与其他结果相联系。5. **交流：** 撰写论文、进行演讲和教学。



图1：研究循环。本文中的系统针对两个重点阶段：问题求解和验证；传统上，这两个阶段是研究中最受关注的部分。

6. **应用：**将其应用于数学中的其他问题以及其他学科。这一阶段还包括数学探索所带来的乐趣。

3.1.2. 其他阐述性著作。Thurston 的随笔《数学中的证明与进步》(1994年)[Thu94] 生动地阐述了理解对数学家意味着什么。我个人也很喜欢 Larry Guth 的随笔《数学对我意味着什么》 [Gut26]。

3.1.3. 超越问题求解。 本笔记中的系统旨在进行问题求解和验证。然而，研究远不止问题求解：优秀的数学家并不等同于优秀的问题求解者。

3.2 为什么研究与竞赛不同

3.2.1. 国际象棋类比与竞赛数学。 数学有时会被比作国际象棋。这个比较相当适用于竞赛（国际数学奥林匹克、普特南数学竞赛等）：考试大纲是固定的；每道题都保证有解；而且有大量历年题目档案可供训练。

3.2.2. 研究有何不同。 然而，研究问题与竞赛题至少有五方面的不同。

1. **开放范围。** 没有固定的行动集合，也无法保证解一定存在，或能通过现有文献中的方法找到。陈述甚至可能无法从标准公理中判定，连续统假设在 ZFC 中便是如此。2. **推广。** 在竞赛中，人们会将以往问题背后的技巧用于解决新问题。而在研究中，解决某个问题的技巧往往无法迁移到其他问题上，新问题可能需要全新的想法。3. **长时域。** 证明可能只有几页，也可能长达数百页；智能体需要数天进行探索，而人类则需要数月甚至数年。一个错误的引理或一条糟糕的路线，都会让此前的一切努力付诸东流。4. **稀疏奖励。** 很难判断某个部分结果或步骤是否有意义，也难以确认一份长篇证明是否完全正确。5. **不断演化且嘈杂的数据。** arXiv 上和期刊中的论文都可能包含错误，而且记录的是完成的证明，而非得出这些证明的推理或直觉，也不包括失败的尝试。

此外，前沿研究按定义并不包含在训练数据中，因此系统必须在测试时学习。

3.2.3. 数学研究因领域而异。不同数学领域的研究也存在很大差异。例如：

- 组合学往往侧重技巧，没有单一的核心主题。
- 几何与范畴朗兰兹纲领往往侧重理论，围绕一个核心主题和一个引领方向的愿景展开。
- 低维拓扑学和辛几何往往侧重直觉；一些几何论证难以严谨地写下来，而可视化至关重要。

无论对人类还是模型而言，在一个领域中的能力是否能迁移到另一个领域，目前尚不明确。然而，各领域彼此交融：一个域中的见解或成果，可能解决另一个领域的问题。

3.2.4. 迁移到其他领域。对于计算机科学家而言，这些特性使数学研究成为测试长时程智能体和推理能力的理想试验场：任务难度很高，但原则上正确性是客观的，而且该领域不受物理世界噪声影响。在这里行之有效的技术很可能也适用于其他长时程任务，例如软件工程、GPU 内核优化和理论物理学中的自动化研究。

4 什么是工具框架和智能体

粗略地说，Harness 工程是运用大语言模型（LLM）来解决特定任务的艺术。

4.1 什么是大语言模型

用一句话来说，大语言模型是一种预测下一个词元的机器，而词元大致相当于一个词。我们先回顾一下什么是词元。

4.1.1. 什么是词元。“词元”一词在不同上下文中有不同含义。在自然语言处理（NLP）和大语言模型（LLM）中，词元是文本的基本单位，通常是一个词或一个词的一部分。例如：

- 苹果。
- 难以置信，的。

还有一些特殊标记，例如 `<EOS>`，它表示一个句子的结束。

4.1.2. 词元在模型训练之前就已确定。 词汇表 $\mathcal{V}$ 由所有词元组成，并在训练之前从文本语料库中学习得到，此后便固定不变。借助词汇表，可以构建一个确定性程序，即 分词器，用于在文本和词元之间进行转换。最受欢迎的选择之一是字节对编码（BPE）分词器；详情请参见[Ras25]。

4.1.3. 大语言模型的三种视角。 理解大语言模型有几种不同的方式：

- 大语言模型是一种通过预测下一个要说的词元来补全句子的机器。
- 大语言模型是一个函数

$$p_\theta \colon \mathcal{V}^{\mathrm{Fin}} \longrightarrow \Delta(\mathcal{V}),$$

其中 $\mathcal{V}^{\mathrm{Fin}}$ 是有限词元序列的集合，而 $\Delta(\mathcal{V})$ 是 $\mathcal{V}$ 上概率分布的集合。它以自回归地方式补全句子：根据输入词元和迄今生成的词元，反复采样下一个词元。

- 大语言模型 p_θ 是一个映射 $\mathcal{V}_{\mathrm{Fin}} \longrightarrow \Delta(\mathcal{V})$，由固定的非线性函数与线性映射复合而成，其中 θ 表示线性映射的参数。训练会更新 θ，以改善模型的预测。

备注 4.1. 大语言模型 是一个固定术语：“大”并不一定意味着模型，也就是它的参数集 θ，规模很大。

4.2 一个初步定义：智能体 = 模型 + 工具框架

定义智能体和工具框架的方式还有好几种。以下是最常见的一种。

4.2.1. 计算机科学中的定义。

定义 4.2. 智能体 =模型 + 工具框架。

用 V. Trivedy（LangChain，2026年）的话来说：“如果你不是模型，那你就是工具框架。”[Tri26]。

备注 4.3. 这个定义方程包含两个未定义的术语。它让我想起牛顿第二定律 $F = ma$（力和质量），这两者在实践中都很有用。这个定义将改进人工智能系统的工作分为两部分：改进模型（通过预训练、中期训练和后训练来更新 θ），以及改进工具框架（θ 保持不变）。

4.2.2. 工具框架让模型能够使用工具。 如图 2所示，模型读取其上下文并生成文本。当输出包含结构化的工具调用时，例如

```
<tool_call>{"name": "search", "query": "quasi-complete local ring"}</tool_call>
```

工具框架执行调用（此处为文献检索），将结果追加到上下文中，然后模型继续生成标记。

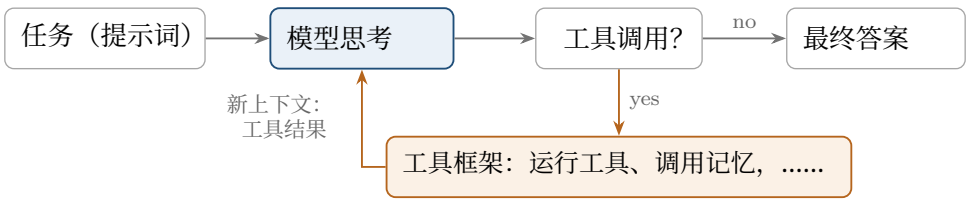

图2：工具使用循环。模型只能看到自己的上下文，而工具框架决定接下来哪些内容进入上下文。

4.3 第二个定义：工具框架 = 上下文管理

4.3.1. 从基本原理出发的定义。归根结底，语言模型是一种用于续写文本的机器。它所做的一切都取决于其上下文窗口。这启发我们从基本原理出发，给出一个实用定义。

定义 4.4。 工具框架是一种上下文管理 系统：一种独立于模型的系统，其目标是在每一步都为模型提供完成任务所需的确切上下文，即所有有用的信息，而不包含任何无关或误导性内容。

4.3.2. 工具框架的“集合论”定义。 工具框架可能包含以下组件。

基础设施： 一个沙箱和一个工作区，智能体可以在其中安全地读写文件并运行程序；进程管理（运行、会话和作业 ID；每个进程的 CPU、记忆、磁盘和挂钟时间限制）；浏览器和计算机使用。

循环支持： 持续运行智能体，直到目标达成；对上下文进行压缩。

工具： 代码执行、Lean、网络搜索等，通常通过模型上下文协议（MCP）提供。

记忆： 纯文本（Markdown 文件）、仅追加的结构化笔记（可搜索的 JSONL 文件）或 SQLite 数据库，并按短期和长期层级进行组织。

提示词和技能： 角色描述（`AGENTS.md`、 `CLAUDE.md`）和技能，即为任务提供指导或流程的 Markdown 指令文件。

编排： 多个智能体如何协作，以及如何在它们之间分配工作。

4.3.3. 所有组件都服务于上下文管理。 工具框架的所有组件（图3）最终都服务于上下文管理。

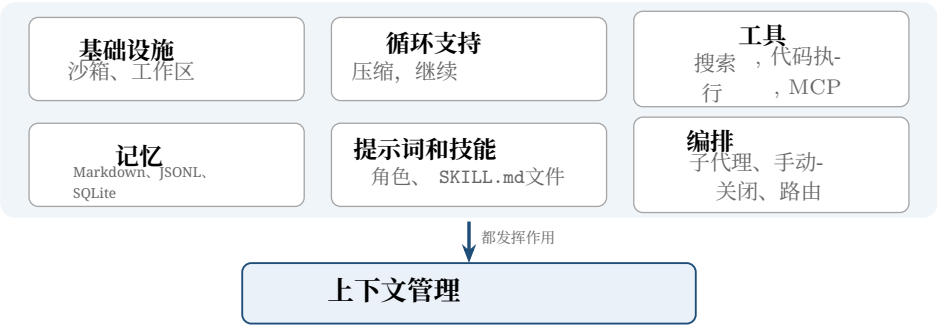

图 3：工具框架组件。

4.4 通用工具框架（Codex、Claude Code）及其局限性

大多数读者已经使用过工具框架。

4.4.1. 前沿模型以通用型工具框架的形式发布。 Claude Fable 5、GPT-6 Astra 及其他前沿模型都以通用型工具框架的形式发布：

- **聊天界面**（ChatGPT Pro、Claude）：扩展思考、网络搜索、代码执行和记忆均在云端管理。

- **编程智能体**（Codex、Claude Code）：它们会在你自己的计算机上运行 Shell 命令，并读取和写入文件，记忆文件也保存在那里。

4.4.2. 内置控制项是一些小型工具框架功能。

- `/goal`：让智能体持续工作，直到达成目标。
- 推理强度最高可达 `max`：每一步都进行更深入的推理。
- `ultra`模式：并行运行更多子代理。

4.4.3. 仍需改进之处。 即使是最强大的模型，在强大的通用工具框架中，至少仍有四个方面需要改进。

1. **数学研究工作流**。 其多智能体编排和流程是为通用目的构建的，而不是为数学研究构建的。2. **上下文工程**。 不相关或具有误导性的信息不应进入上下文。3. **长时域**。 当一个上下文中包含多条工作线（主题 A，然后是主题 B）时，模型会忘记自己正在推进哪一条。4. **计算资源的高效利用**。 经过更好工程设计的工具框架，比 `/goal` 或 `ultra` 更有效地利用计算资源，也能更好地与人类标准保持一致，从而同时节省资金和人类精力。

4.4.4. 总结。 只要任务超出上下文窗口，或上下文混淆了模型注意力，工具框架就至少会发挥重要作用。

4.5 为什么研究工具框架和智能体

4.5.1. 为什么研究工具框架，而不是模型训练。

1. **计算机科学方面的原因**。 对于大多数任务，应先尝试免训练方法，再进行训练。与模型相比，工具框架成本更低，迭代也更快。此外，有效的工具框架可以通过生成轨迹作为合成训练数据，或通过智能体强化学习，来助力训练。2. **经济方面的原因**。 对大多数研究团队而言，训练具备前沿数学能力的模型并不可行，因为成本极高。有效的工具框架能够更高效地扩展计算资源，并在相同预算下更充分地发挥模型的作用。它还可能让成本更低的模型或开源模型胜任某些任务。3. **其他原因**。 优秀的工具框架依赖于对基础模型优势和弱点的理解，而设计工具框架也能让我们形成关于基础模型应如何改进的直觉。它还可能加深我们对任务本身的高层次理解。

4.5.2. 工具框架会随着基础模型的改进而演进。 随着模型不断改进，它们会吸收工具框架中较简单的部分，并使更具挑战性的部分成为可能。为较弱模型编写的逐步技能指令，对更强的模型而言可能并无必要；更强的模型可以转而利用为更困难任务构建的工具框架，例如递归自我改进。

4.5.3. 哪些组件在更强大的模型出现后仍能发挥作用？弥补模型弱点的组件（例如详细的技能或检索技巧）往往会随着模型改进而变得不再必要；例如，Pharos 就移除了 Danus 的好几个此类组件。编码结构或管理上下文的组件（例如为每个智能体设置有界上下文，或分离角色）显然并不取决于模型的强弱。只要任务的长度超过上下文窗口，就需要某种工具框架。

4.6 设计原则：如何设计工具框架和智能体

4.6.1. 超越反复试错。设计有效的工具框架在很大程度上依赖实证，也需要反复试错。不过，有几项原则可以指导设计。第 5 节和第 6 节中的系统就是依据以下三项原则设计的。

4.6.2. 第一性原理：上下文工程。

原则 1. 提供正确的上下文：包括模型解题所需的一切内容，不包含任何无关信息。

例如，验证智能体应从全新上下文开始，其中不包含解的推理轨迹；低质量信息则应精简剔除。

4.6.3. 数学直觉：模仿人类工作流程。

原则 2. 模仿人类专家解决任务的方式。

例如，将人类的习惯提炼为技能，将外部资源转化为工具，并将笔记本变成记忆。Danus 和 Pharos 的多智能体编排模仿了研究团队。

4.6.4. 计算机科学直觉：有意义地扩展计算资源。

原则 3. 扩展计算规模而不造成太多浪费，计算资源就能换来智能。

我们希望，10 位智商为 140 的聪明人经过精心设计的协作，能够比肩一位智商为 400 的人。

用于证明和问题求解的 5 个智能体：Rethlas、Danus、Pharos

本节将介绍三个用于证明或反驳猜想的智能体：Rethlas、Danus 和 Pharos。对于每个智能体，我们先比较仅使用基础模型与使用工具框架时的结果，然后介绍其设计，详细考察案例研究，最后总结促使我们开发下一个系统的局限性。

5.1 Rethlas：生成–验证循环

Rethlas 完全开源，其技术报告见 [Ju+26]，代码见 https://github.com/frenzymath/Rethlas。

5.1.1 与基础模型的比较

5.1.1.1. 比较。 表 3 比较了同一基础模型在有无 Rethlas框架的情况下处理三个问题的表现。每种情况下，提供给 Rethlas 的只有问题陈述。

问题	域	仅使用模型	借助 Rethlas
……的有理共轭余特征 [Ju+26，§5.1。1]	代数 群 组合学	GPT-5.5 Pro：错误证明 GPT-5.6 Sol Ultra：不完整	GPT-5.5：正确的证明，44 分钟 GPT-5.6 Sol：反例，4 小时 59 分钟
拟阵 KL 多项式的单峰性多项式 [CL26]			
Manin 常数猜想 [CLN26]	代数 数论	GPT-6 Astra：需要多轮人工指导	GPT-6 Astra，使用 OSVerify 作为验证器：正确的证明，完全自动化

表3：同一模型在使用和不使用工具框架时的表现。“仅模型”指的是该模型在最强设置下的网页端界面。

5.1.1.2. 其他结果。 目前正在此处维护一份使用 Rethlas 获得的、不断扩充的部分结果列表 https://github.com/frenzymath/Rethlas_results。例如：

- 交换代数：Anderson 的问题 [Ju+26，§4]；Cahen–Fontana–Frisch–Glaz 问题集中的八个问题 [CFFG14；Jia+26]；Erman–Sam 关于 Boij–Söderberg 理论的问题；
- p-进 *Hodge* 理论：精化的提升无关性定理 [PY26]；
- 分析学与概率论：Brezis 列表中的两个问题 [DJ26]；相位恢复的单射性；
- 组合学：拟阵的反 Kazhdan–Lusztig 多项式不一定是对数凹的；无环拟阵的反 Z 多项式具有单峰系数。
- 相对几何朗兰兹（以 OSVerify 为验证器）：自守交换子关系 [Liu26a]；Plancherel 代数的性质 [Liu26b]。

5.1.2 智能体工作流设计

Rethlas（图 4）由两个智能体组成。

5.1.2.1. 生成。 **生成代理**撰写一份候选证明，称为蓝图：一份类似论文的 Markdown 文件，包含若干引理，每条引理均附有陈述和证明，最后以原样呈现的主定理收尾。

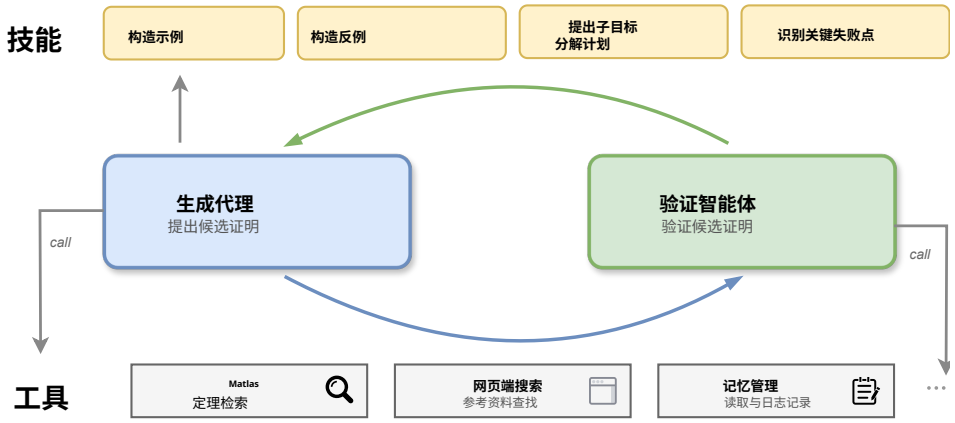

图4：Rethlas 的工作流设计 [Ju+26]。

5.1.2.2. 验证。 **验证智能体**审阅该蓝图。每份提交都会由一名拥有全新上下文的智能体评判，并返回一份结构化报告：包括摘要、关键错误列表、缺漏列表和修复提示。

5.1.2.3. 迭代。生成-验证循环会不断重复，直到接受为止；接受要求不存在关键错误和缺漏。

5.1.3 Rethlas 中的设计原则

该设计严格遵循第 4.6 节的原则。

5.1.3.1. 模仿人类工作流程：技能即习惯。Rethlas 将数学家解决难题时的一些习惯编码为技能：生成代理在判断相关时加载的 Markdown 指令文件。

探索	构造玩具例子；构造反例； 搜索相关结果；查询记忆
规划	提出子目标分解计划
证明	获得直接结论；直接证明；递归证明
迭代	识别关键失败原因；验证证明

没有固定顺序：智能体会循环执行评估状态、选择技能、采取行动。在递归证明中，彼此竞争的分解计划作为子代理并行运行，每个子代理都会获知所有计划中发现的困难；如果全部失败，识别关键失败原因 技能会提取它们共同的障碍，为下一轮计划提供种子。

下面是一个技能示例：

construct-toy-examples

```
## Procedure
1. Construct simpler cases (low degree, small dimension, special forms,
 canonical objects).
2. Ensure the toy example satisfies all assumptions of
the target statement or subgoal.
3. Check that the conclusion also holds in the toy example.
```

验证器拥有自己的技能。它会检查最小推理步骤的有效性、适用性以及是否缺少假设，并检查引用是否正确：外部结果必须附上完整的陈述和来源。

5.1.3.2. 模仿人类工作流：搜索文献，并独立思考。 核心工具是 **Matlas**（https://matlas.ai），它是一个针对数学陈述的语义搜索引擎。它由来自 180 种期刊的 43.5 万篇已发表论文和 1,900 多本教科书中的 851 万条陈述构建而成，并通过依赖关系提取，使检索到的陈述能够自成一体[Ju+26]。Rethlas 在奇数次迭代中禁用搜索，在偶数次迭代中重新启用搜索，以平衡 深入思考 与从文献中学习。

5.1.3.3. 模仿人类工作流：笔记本就是记忆。 原始日志是草稿纸。结构化笔记按问题分别保存在只追加的 JSONL 通道中（`toy_examples`,`counterexamples`, `failed_paths`, `verification_reports`, ...），并通过 BM25 搜索检索。

5.1.3.4. 模仿人类工作流：作者不是审稿人。 作者相信自己的证明，模型也一样；自我检查只能在有限程度上减少错误。数学界的应对方式是审稿人——一个对证明是否成立毫无利害关系的人。

5.1.3.5. 上下文工程：持续的作者，全新的审稿人。

- 生成器拥有持续的上下文，因此直觉、部分进展和死胡同都会在迭代中延续；
- 验证器在每次提交时都会获得全新的上下文：它只能看到蓝图和参考文献，不知道蓝图是如何生成的，因此不会受到推理轨迹的干扰。

5.1.3.6. 有效扩展计算规模。 生成代理会搜索候选解。由于失败路径会被记录，计算资源不会浪费在重复过去的错误上。

5.1.4 案例研究：版权与理解

这份技术报告 [Ju+26] 包含三个案例研究：D. D. Anderson 在交换代数中的一个开放问题、一个关于 p-adic Hodge 理论 [PY26], 的问题，以及下文讨论的代数群问题。报告分析了 Rethlas 的推理轨迹；而本文讨论的是我们观察到的两个现象。

5.1.4.1. 案例研究 *1*：幻觉还是天才式妙招？提示词如下：一道代数群问题。

问题 5.1 (B. Conrad 提出)。 设 G 是域 k 上的一个连通光滑仿射群，且 $\mathbb{X}_*(G) = \mathrm{Hom}^{k\text{-}gp}(\mathrm{GL}_1, G)$。对于一个域扩张 K/k，映射 $\mathbb{X}_*(G)/G(k) \to \mathbb{X}_*(G_K)/G(K)$ 是单射吗？

约化 G 和完美 k情形的答案已为人所知；一般情形尚未见于文献。Rethlas 和 GPT-5.5 Pro 都解决了简单情形。对于一般情形，GPT-5.5 Pro（网页端）给出了错误的证明，没有取得任何实质性进展，而 Rethlas（GPT-5.5 `xhigh`）在 44 分钟内证明了单射性。

其策略是先约化到拟约化情形（该智能体创造了“扭曲幂零情形”这一术语），然后比较相对根与绝对根以及魏尔群。

不过，这里有一个小错误。证明需要一个关键要素，[CGP15，定理 C.2。15]，该定理来自一本受版权保护的书。Rethlas 要求智能体完整引用其所援引的每一项外部结果的陈述，而在这里，这就意味着必须复制受版权保护的文本。因此，智能体没有引用 [CGP15]。相反，它在网页端搜索了 `CGP Thm C.2.15`，找到了综述[CP17]，该综述位于 [CGP15] 的 Conrad 主页上，并引用了 [CP17，定理 5.3.2(iv)]。该定理的证明回溯引用了 [CGP15，定理 C.2，15]，但其中的陈述并不完整，也没有包含证明所需的断言。

5.1.4.2. 案例研究 *2*：智能体能提取出方法吗？本案例研究表明，理解仍然需要人类专家：智能体恰好解决了交给它的任务，但其解背后的通用方法必须由人类提取出来。

拟阵理论中有两个猜想：

猜想 5.2 (Gedeon–Proudfoot–Young 2017)。拟阵的 ***Kazhdan–Lusztig*** 多项式 $P_M(t)$ M 是实根。

猜想 5.3 (Elias–Proudfoot–Wakefield 2016)。拟阵的 ***Kazhdan–Lusztig*** 多项式 $P_M(t)$ M 是对数凹的。

我们只用 Rethlas 尝试了第一个猜想。给 Rethlas 的提示词只包含 P_M 的定义，以及指令：“证明或证伪 $P_M(t)$ 的所有零点都位于负实轴上。”Rethlas（GPT-5.6 Sol，`max`）在 4 小时 59 分钟内找到了一个反驳猜想 5.2 的显式反例。通过网页端界面使用两个提示词的 GPT-5.6 Sol Ultra 则未能完成。

然而，智能体只完成了所分配任务所需的最低限度工作。人类作者 [CL26]将其构造转化为一种方法（[CL26，定理 1。1]），赋予它几何解释（[CL26，§3]），并用它证明 P_M 甚至不必是单峰，从而一举推翻猜想5.2、5.3 。智能体并未推广其构造，也没有从几何角度解释它，或注意到它与第二个猜想有关。

5.1.5 为什么生成—验证循环还不够

单一循环有若干局限性，这促使我们转向接下来的系统。

5.1.5.1. 缺乏问题分解。数学家会积累许多小事实，并随着新事实不断出现而调整策略。Rethlas 一次只研究一个假设，就像单一的深度优先搜索，无法融合不同领域的专长。

5.1.5.2. 线性逻辑结构。一个问题可能需要多人沿不同路径开展工作，而这些路径又彼此交互，因此工作的逻辑结构未必是线性的。单个 Markdown 蓝图无法容纳这种结构，而且两个智能体无法同时安全地编辑它，因此不能简单地将循环并行化。

5.1.5.3. 上下文工程。每一轮，生成器都会重新阅读整份草稿，包括所有经过验证的部分结果及其证明，即使只需要陈述，而引理应当作为黑箱使用。

5.1.5.4. 通过扩展计算规模来获取智能。 Rethlas 包含两个组件：生成和验证，二者都可以扩展。Danus（第 5.2 节）和 Pharos（第 5.3 节）扩展生成所使用的计算资源，从单个作者—审稿人循环扩展到研究团队。OSVerify（第6 节）扩展验证所使用的计算资源，以获得更严格的验证器。

5.2 Danus：智能体研究组

Danus 完全开源，论文见 [Liu+26]，代码见 https://github.com/frenzymath/Danus。GPT-6 Astra 发布后，Danus 已被 Pharos 取代，但在使用开源模型或较弱模型时仍可能有用。

5.2.1 设计

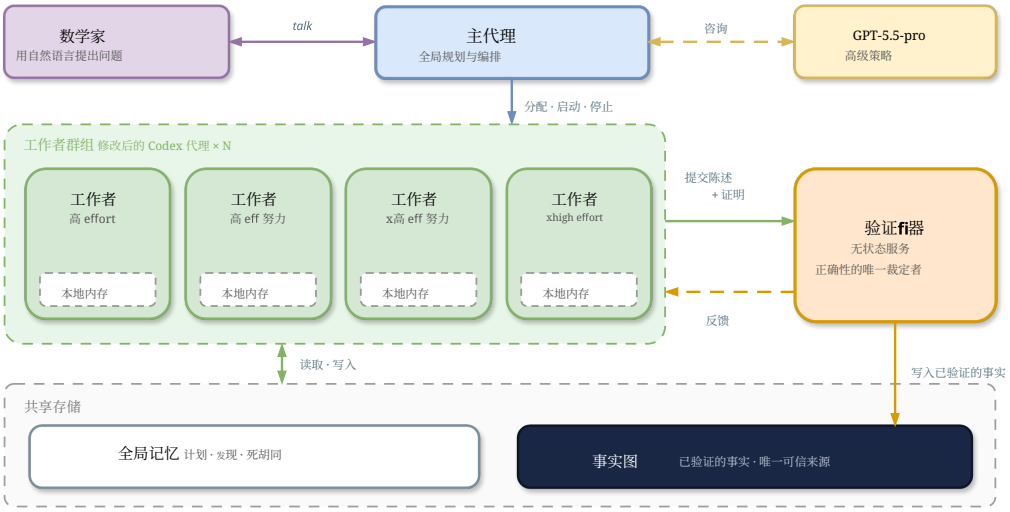

图 5：Danus 架构 [Liu+26]。

5.2.1.1. 智能体工作流。 Danus 模拟一个研究团队（图 5）。

- **主代理** 是团队负责人（在论文实验中使用的是搭载 Opus 4.8 的 Claude Code）。它规划路径、分配并监控工作，并向人类汇报。它不进行数学工作。
- **高级专家** （GPT-5.5 Pro）每小时至多接受一次咨询，咨询内容是对状态的提炼摘要。
- **验证器** 是无状态的：每个提交的主张都由一个全新的实例进行判断。
- **工作智能体** 是 Rethlas 生成智能体（搭载 GPT-5.5 的 Codex），每个项目有三到九个。每个工作智能体一次负责一个小问题，由主代理分配。

5.2.1.2. 事实图。 交付成果不再是单一文档，而是一个有向无环图，其节点是事实 （包含一项陈述及其证明，且该证明已被验证器接受），边则表示逻辑依赖关系。

5.2.1.3. 为何需要事实图？ 事实图有以下几种用途：

1. 它将分解问题为小步骤，使每个工作智能体面对更简单的任务；2. 它是工作智能体之间传递经验证结果的交流渠道；

3. 它提供了更清晰的上下文：工作智能体仅按陈述加载所需的事实，只有在需要某种技术时才读取证明；

4. 错误　处更容易修正：撤销一个事实会将其删除，并删除所有依赖于它的事实　t；

5. 它可扩展生成时的计算资源：每个工作者与无状态验证器共同构成一个 Rethlas 循环（图 6），而且由于交付物是模块化的，并行编辑不会相互冲突。

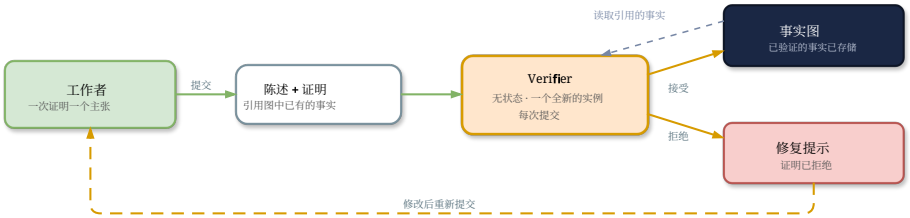

图 6：Danus 内部的一个 Rethlas 循环。

5.2.1.4. 三层记忆。 Danus 将真值与认知分开：

Tier	目录	状态	人类类比
事实图	已验证事实及其依赖关系	唯一的真值	该群体已验证的草稿（例如 Overleaf 项目）
全局记忆	计划、示例、反例、已失效的 ends	共享的，不 真值	在其中分享的想法
本地内存	每个工作者的笔记和日志	草稿纸	个人笔记

5.2.1.5. 每个边界处的压缩。 原始工作者日志会被压缩到全局记忆中，全局记忆再被压缩到主代理的检查点摘要中，最后这些摘要被整理成供人类阅读的报告。工作者只能看到自己的任务、经过压缩的全局记忆，以及从事实图中检索到的事实。任何单个智能体所读取的内容都保持在有限范围内。

5.2.1.6. 通过工具强制执行的权力分立。在大规模情况下，角色分工会演变为权力分立。在 Danus 中，这种分立是通过每个角色可以调用的 MCP 工具来强制执行的，而不是通过 Markdown 文件中的指令来实现：

	main	工作者	验证器
读取事实	✓	✓	✓
写入记忆	✓	✓	×
提交事实（`fact_submit`）	×	✓	×
判定真值	×	×	✓

特别是，主代理无法向事实图提交事实。

5.2.1.7. 在宽度和深度两个维度扩展生成。宽度：事实图将证明拆分为工作者可以负责的单元，使贡献得以累积，同时主代理可以探索多样化的方向。深度：逐事实验证和记忆层级将推理跨度扩展到单个上下文窗口之外。

5.2.1.8. 完成与论文撰写。 当原始问题陈述进入事实图时，Danus 便宣布任务成功。随后，主代理根据事实图撰写论文并提交给验证器；这一撰写—验证循环会持续进行，直到验证器接受论文。

5.2.2 案例研究：拟阵构造集的切丛类

这篇论文 [Liu+26] 包含六个案例研究；这里我们将详细讨论其中一个，即拟阵构造集的切丛类。

5.2.2.1. 问题陈述。 设 M 是一个秩为 $d+1$ 的无环拟阵（不一定可实现），且 $\mathcal{G}$ 是一个构造集。构造组合 K-环 $K(M,\mathcal{G})$ 中的一个类 $T_{M,\mathcal{G}}$，使得：(i) 若 M 可实现，则 $T_{M,\mathcal{G}}$ 是精彩紧化的切丛类；(ii) Hirzebruch–Riemann–Roch 公式 $\dim A^i(M,\mathcal{G}) = (-1)^i\chi(\wedge^i T^\vee)$ 成立；且 (iii) 陈数满足 $\deg(c_k(T)\alpha^{d-k}) \geq \binom{d+1}{k}$。

这是 Cheng 论文的主要结果 [Che26],独立得出。

5.2.2.2. 与基础模型的比较。 使用完全相同的原样提示词，GPT-5.5 Pro（网页端）毫无进展，也没有策略计划；Rethlas 运行了三次，生成了证明草稿，但都未通过内部验证智能体（0/3）；Danus 则得到了经过验证的解 [CLG26]。

5.2.2.3. 我们如何选定这个问题。 纯属巧合。2026年5月，我们在 California Avenue 共进午餐时，Ronnie Cheng 告诉我，他刚完成了论文 [Che26], ，而 GPT-5.5 Pro 无法证明其主要结果。他慷慨地同意让我们尝试用自己的系统解决这些问题，并推迟了将论文发布到 arXiv 的时间。GPT-5.5 Pro 和 Rethlas 都失败了；Danus 重新证明了主要结果 [CLG26]。

5.2.2.4. 实验设置。 本次运行使用了 7 个工作智能体（3 个以 `xhigh` 努力度运行，4 个以 `high` 努力度运行）。整个过程约耗时五天，其中活跃时长约 50 小时，并进行了 13 次 GPT-5.5 Pro 咨询。

5.2.2.5. *Danus* 可扩展生成的宽度和深度。 事实图（图 7）包含 3,157 条已验证事实和 8,616 条依赖边，事实链最长可达 54 条。主定理只需要 499 条事实，因此证明出的事实数量约为最终证明所用事实的六倍。

5.2.2.6. 计算资源可以换取智能。 Rethlas 运行的日志显示，Rethlas 混淆了几个相近的概念（可实现拟阵、奇妙模型、环面簇；积分理论与有理K-理论）。这表明，数学上的细微之处，例如相近概念之间的差异，可以通过增加计算资源的宽度和深度来克服：思考得更久、更仔细。

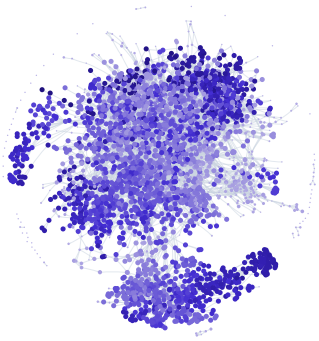

图7：切线类运行的事实图 [Liu+26]。

5.2.3 局限性

Danus 有若干局限性。

- 验证器偶尔会出现误报。一旦生成所使用的计算资源得到扩展，验证所使用的计算资源也必须相应扩展，以免事实图中的错误破坏整个运行过程。这促成了 OSVerify（第6节）。

- 会被更强大的基础模型淘汰。当基础模型得到改进后，应为每个工作者分配更困难的任务，而不是细小的步骤。这促成了 Pharos（第5.3节）。
- 与人类专家的写作和交流。论文和报告对人类而言很难快速阅读，因为其中的步骤过度压缩、使用了非常规记号，等等。

5.3 Pharos：面向更强模型的 Danus

Pharos 完全开源，代码位于 https://github.com/frenzymath/Pharos，技术报告见 https://frenzymath.com/news/pharos-preview/，另有一篇正在撰写的论文[Gao+26]；另请参阅 [Gao26]。

5.3.1 设计

5.3.1.1. 与 *Danus* 的比较概述。 Pharos 沿用了 Danus 的事实图、工作智能体和验证器，但针对 GPT-6 Astra 及其他更强大的模型重新构建了其余部分。一些用于弥补较弱模型不足的组件已被移除。表 4 列出了部分改动。

5.3.1.2. 路由板。 主要新增结构是一个路由板（文件 `ROUTES.md`），这是主代理实时记录策略和计划的地方。图8 展示了更新周期。

已知路径 **首先**。 在构思新路径之前，会先尝试文献中的每一种技术。

实时状态。 看板显示当前状态以及下一步的尝试方向。这解决了第 4.4.3节中的长上下文混淆：每个工作者只能看到自己路线的简报。

	Danus（2026年7月）	**Pharos v3.1**（2026年9月）
主代理	Claude Code，Opus 4.8	Codex， ultra努力程度
顾问	GPT-5.5 Pro 每隔 1–2小时	顾问已移除
规划	定期摘要	路由板
写作	一个写作-验证循环	大纲 → 逐节写作者-验证者对 → 整篇论文验证器；提炼的专家写作风格
计算	没有大型数值计算	允许，并设置计算上限和进度监视器
工作智能体	逐轮节奏	围绕一项结果开展多轮工作
记忆	记忆的完整扫描	用于记忆搜索的增量索引
人工输入	临时的	逐字记录
模型	GPT-5.5 工作者	GPT-6 Astra 工作者

表4：从 Danus 到 Pharos。

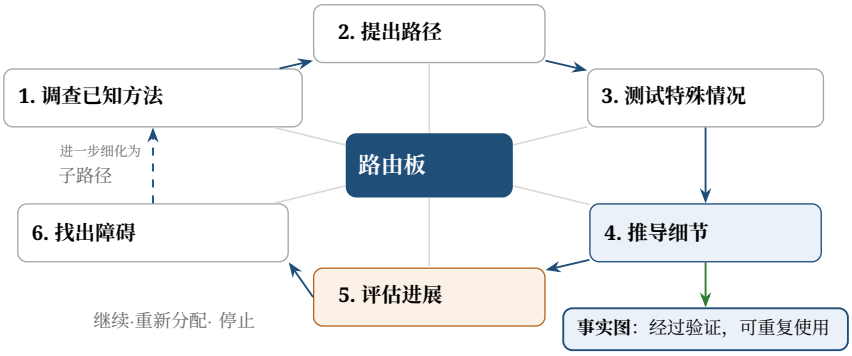

图8：Pharos 路由板的更新方式 [Gao26]。

5.3.1.3. 其他改进。

- **更深入的事实**。 Danus 工作者处理每条小型事实大约需要十分钟；Pharos 工作者处理较难事实则需要一到两小时，这适合 GPT-6 Astra。
- **辅助智能体** 梳理文献、探查路线并评估进展。
- **计算**。 工作智能体在预算上限内运行代码并进行数值实验。
- **与专家协作**。专家可以跟进一次运行，并通过主代理引导其推进。
- **效率**。无损上下文压缩可降低成本并提高吞吐量。
- **改进后的写作流程**。一旦目标确定为某个事实，主代理就会撰写提纲并分配各章节；每位章节撰写者都会根据各自的验证器进行修订；整篇论文验证器会检查汇总后的论证；随后对论文进行润色并生成 PDF。

5.3.1.4. 新角色：运维代理。此实现会在代码仓库根目录添加一个运维代理，用于部署和监控多个项目，并实现用户定制。

5.3.2 案例研究 1：Hartshorne 猜想的代数证明

第一个案例研究是关于 Miyaoka–Mori 判据的代数特征零证明，该证明重新证明了 Hartshorne 猜想 [唐+26]。文中在 [唐+26附录 B]中介绍了如何使用 Pharos。

5.3.2.1. 问题陈述。 对于特征为零的光滑射影簇 X 和一条曲线f： $C \to X$，在严格的次数界限下， $-K_X \cdot f_*[C] > 0$ 是否单独足以保证存在一条穿过 $f(C)$ 中每一点的有理曲线?

5.3.2.2. 与基础模型的比较。 Du、Guo 和 Xie 在网页端尝试了 GPT-6 Astra Pro，但即使经过多轮数学指导，也未能获得解。相比之下，Pharos 搭配 GPT-6 Astra 完全自动地解决了该问题，用时 19 小时 57 分钟，无需人类干预 [Don+26]。

5.3.2.3. 实验设置。 主代理以 `ultra` 努力运行，3 个工作智能体以 `xhigh` 运行，4 个以 `max`运行。总成本不到每周 Pro 使用限额的两倍。

5.3.2.4. 更深入的事实与探索。 事实图中有 273 条已验证事实（其中一条曾被撤销后重新证明），但最终证明中仅用到 4 条。

5.3.2.5. 不同的路径及其相互作用。 共有 7 条路径，细分为 30 条子路径，各条路径之间会重复使用事实。最终证明通过一条新的子路径 R3-6，将路径 R3 和 R6 结合起来。

5.3.2.6. 人类提示可以节省计算资源。 启动时提供专家提示后，Pharos 在 11 小时 38 分钟内找到了同一个证明。

5.3.3 案例研究 2：创建一个框架，以统一两个问题

本案例研究由 A. Shelley 和 H. Bosch 使用 Pharos 完成 [SB26]。该*loc.cit*引用了 Danus，因为当时 Pharos 尚未命名。

5.3.3.1. 背景。 Shelley 针对两个问题提出了若干相关下界：静止测地线网络和牛顿点弹球。Bosch 在其关于静止测地线网络的研究中证明了一个超指数界 [Bos26]。他们希望从共同的角度审视这两个问题，并优化相关构造。

5.3.3.2. 从一个界到统一框架。 首次尝试时，Pharos 并未改进构造。拿到 Bosch 的论文和新的下界后，它花了大约一个小时，找到了一份证明，将 Bosch 的界从超指数级改进为指数级。随后，Bosch 提出这两个问题或许可以纳入一个统一框架。收到这一提示词后，Pharos 花了半小时到一小时，写出一份手稿，以一个共同定义统一了这两个问题，并采用了哈密顿力学的形式体系。碰撞系统 这个名称出自一位人类；而使该框架行之有效的定义则出自智能体。

5.3.3.3. 优化上下界。 接下来，在受命优化上下界后，Pharos 将最优基数的下界从 $2^{1/3}$ 提高到 $3^{1/4}$ （在对问题作出放宽的条件下），并将上界从 8 降至3。98。随后，作者们与 Claude Fable 一起花了几天时间完善证明，降低其复杂度并提升其清晰度，然后将其发布在 Hexagon [SB26]。

5.3.3.4. 与基础模型的比较。据作者称，Claude Fable 5.1单独使用时，未能为这两个问题找到统一框架。

6 用于验证的智能体：OSVerify

OSVerify 完全开源，其论文 [Nie+26]、网页端位于 https://osverify.github.io，代码位于 https://github.com/OSVerify。

6.1 动机与目标

6.1.1. 面向人类专家。验证已成为成本高昂的环节。如今，生成论文的速度远快于评审速度，因此我们需要减少审稿所耗费的人力。

6.1.2. 面向推理智能体。 一旦智能体系统扩大生成规模（Danus 和 Pharos 就是如此），薄弱的验证器就会毁掉整个运行过程：错误引理会在事实图中堆积，如同遭受拒绝服务攻击。因此，测试时计算也应分配给验证器。

6.1.3. 目标：一位更严格的评审者。 我们希望有一个更严格的验证器，能够可靠地拒绝错误的解答，同时很少拒绝正确的解答。形式化地说，验证器 $V(x,\ y) \in \{\text{拒绝，通过}\}$ 对问题 x 的解答 y 作出判断，而我们希望它在以下意义上是单侧的。

定义 6.1 （单侧验证）。 在假阴性率 FNR $\leq \tau$ 不超过较小容差 τ 的前提下，最大化真阴性率 TNR，其中

$$\mathrm{TNR} = \Pr[\textsc{Reject} \mid y \text{ invalid}], \qquad \mathrm{FNR} = \Pr[\textsc{Reject} \mid y \text{ valid}].$$

6.2 设计：提出—验证循环

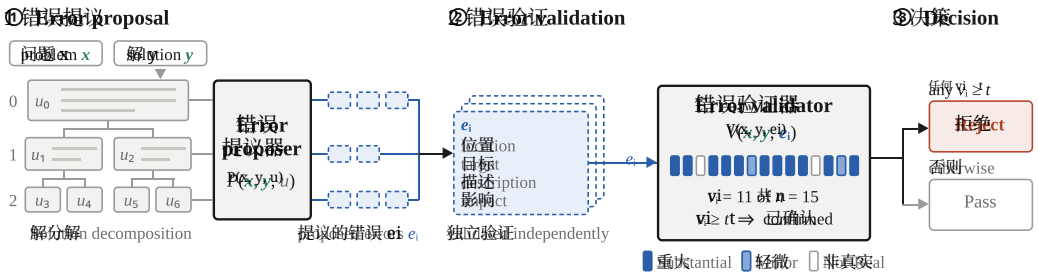

图 9：OSVerify [Nie+26]：错误提议器将解分解并提出错误；错误验证器则判断每个提出的错误。

6.2.1. 智能体工作流。 OSVerify 由两个智能体组成（图9）。

错误提议器（目标：覆盖率）。以层次化方式分解解，并提出候选错误。

错误验证器（目标：不产生误报）。将每个候选错误分类为非真实、轻微或重大。

6.2.2. 设计依据。以下几项观察促成了这一设计。

- 模仿人类工作流程。人工裁判首先怀疑某个步骤，然后检查它是否构成真实且致命的错误。OSVerify 为每项任务分配一个独立的智能体。
- 错误比发现更容易确认。即使大语言模型未能定位推理错误，告知错误所在位置后，它们通常也能将其纠正 [TMCCM24]。
- 上下文工程。提议器一次检查一个部分，同时保留完整的解；每次验证器调用都集中处理单个错误。
- 扩展计算规模。 Rethlas 的生成—验证原则被应用于验证之中。由此产生的二阶生成—验证循环扩展了计算规模，并产生了更严格的审稿人。

6.3 与 LLM 评审和多数投票的比较

6.3.1. 基准测试。基准测试包括 Putnam100（58 个无效解答和 50 个有效解答）、Hard2Verify（H2V） [Pan+26]（163 个无效解答，37 个有效解答），以及 FirstProof-Eval，该基准由二十道 First Proof 问题构建而成 [Abo+26]（59 个无效解答，35 个有效解答）。H2V 和 First Proof 均在这两个骨干模型之后发布。

6.3.2. 骨干模型。 这些骨干模型刻意选择得较弱（`gpt-oss-120b` 和 `gpt-5-mini`），以模拟超出模型可靠能力范围的问题。

6.3.3. 训练验证器。 验证器也可以通过强化学习（使用 LoRA 的 GRPO）在带标签的（问题、解答、错误）三元组数据集上进行训练；该数据集中未使用 First Proof。详情请参阅 [Nie+26，附录 D]。

6.3.4. 结果。 表 5比较了 OSVerify、LLM 评审和多数投票。

方法 (`gpt-oss-120b`)	**Putnam100**	**H2V**	**FirstProof-Eval**
LLM 评审	0	54.4	0
投票（16 个样本）	38.4	69.2	35.9
OSVerify	70.4	**84.7**	58.0
+ 强化学习训练的验证器	**89.6**	83.2	**62.3**

表 5：在 假阴性率 $\leq 5\%$ [Nie+26] 时的真阴性率 0 表示只有始终通过规则满足约束。

6.3.5. 扩展提议器和验证器。 这两个组件都能从更多计算资源中受益（图 10）。更深入的分解可提高真阴性率，大部分提升在深度为二时已实现。增加验证器投票次数也能提高真阴性率，而更严格的假阴性率约束则需要更多投票。在令牌预算相同的情况下，OSVerify 始终优于投票；强化学习训练的验证器只需约四分之一的计算资源，就能达到相同的真阴性率。

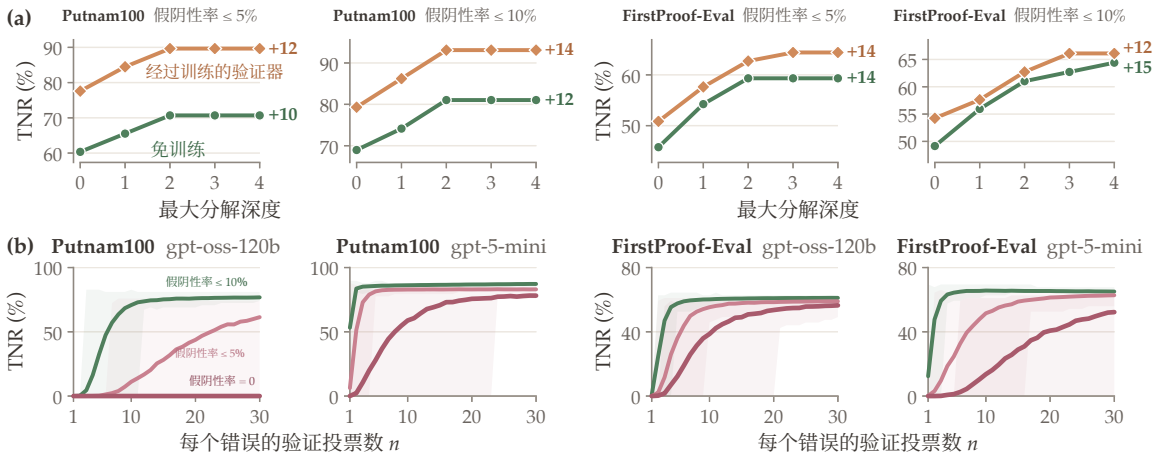

图10：Sc 按验证进行计算资源扩展 [Nie+26]：（a）提议器深度；（b）验证器投票数 .

6.4 与前沿模型的比较：数学研究案例研究

借助前沿骨干模型，OSVerify 已被用于评审研究论文。

6.4.1. 人工智能生成的论文。 OpenAI 发布了十一篇预印本，每篇都声称在一个核心问题上取得了进展（四维 Kakeya 问题、平面单位距离、三线性希尔伯特变换、量子几何朗兰兹，等等）。OSVerify 在每篇论文中都发现了一处实质性错误。这些错误被 OpenAI 的内部审查漏过，并经专家确认，尽管经过修订后，主要结论仍然成立。参见 https://osverify.github.io/case-studies.html。

6.4.2. *Kamnitzer* 的 *2022* 年国际数学家大会猜想。 Q. Tang 撰写了一篇人工智能辅助撰写的手稿，证明 Mirković–Vilonen 基与 theta 基相同 [Tan26]。第一轮中，GPT-6 Pro 起草了证明，Codex 对其进行了评审；OSVerify 发现了一处重大错误。第二轮中，GPT-6 Pro 修正了证明，ChatGPT 和 Codex 又进行了三次评审，均未发现问题；OSVerify 则发现了另一处足以推翻证明的错误。

6.4.3. *Manin* 常数猜想。 Z. Chi、S. Liu 和 P. Nie 获得了关于 Manin 常数猜想的人工智能辅助证明 [CLN26]，该猜想源于对 Birch 和 Swinnerton-Dyer 猜想的研究，此前已悬而未决 55 年以上。对早期草稿进行了十二轮相互独立的人工智能检查（Claude Opus 5.5、Claude Fable 5.1，以及搭载 GPT-6 Astra 的 Codex），均未发现错误。OSVerify（搭载 GPT-5.6 Sol）发现了一处错误，并针对其影响的三个位置提出了修正建议。参见 [CLN26，附录 A]。

6.5 OSVerify 作为元算法

6.5.1. 用于验证。 任何能够指出错误的验证器，例如多智能体辩论 [DLTTM24] 或伪形式化 [BBWAM 26], 都可以充当提议器，并在其上叠加 OSVerify 验证器。在 H2V 上，当假阴性率为 0% 时，这使多智能体辩论的真阴性率从 71.2% 提高到80.8%；而伪形式化的真阴性率则从0 提高到 92.6%。

6.5.2. 用于问题求解。 OSVerify 可以替代 Rethlas、Danus 和 Pharos 等问题求解智能体中的简单验证器；Rethlas-OSV（第 7 节）就是一个例子。

7 如何以数学家的身份使用和贡献

本节首先介绍如何使用本文介绍的智能体，然后说明如何为改进它们作出贡献。

这些智能体是为 Linux 和 macOS 设计的，以下说明也针对这两个系统。Windows 用户可以使用 WSL 或虚拟机，并向编程智能体寻求帮助，但不保证性能。

7.1 选择工具：别拿大锤砸核桃

7.1.1. 经验建议。 表 6 给出了作者截至2026年11月的经验估算。运行时间和配额会因问题和模型而有很大差异。

系统	适用场景	典型运行时间
聊天（GPT Pro，网页端）	讨论、小问题、文献	分钟
Codex / Claude Code (`/goal`)	一个你预计可以解决的问题	30 分钟–3 小时
Rethlas	一个解可能很简单的问题 逻辑结构	1–5 小时
Danus / Pharos	一个需要许多 引理的复杂问题	约每周一个 Pro 每 12 小时的配额
OSVerify	检查论文或证明	1–5 小时（每周配额的 <10%）
Rethlas + OSVerify	使用更严格的验证器求解	1–5 小时（占每周配额的 <20%（每周配额，使用 1 个分支）

表6：不同任务适用的工具。

7.1.2. *Danus* 还是 *Pharos*。对于 GPT-6 Astra、GPT-6.1 Sol 或 Claude Opus 5.5 等更强大的模型，强烈建议使用 Pharos：据观察，Danus 会限制更强大模型的性能。对于 GPT-5.5 或开源模型等较弱模型，Danus 可能是更好的选择，而强大的模型可以充当其外部顾问代理。

7.1.3. 将 *OSVerify* 用作验证器。Pharos 也可以与 OSVerify 结合使用，但成本可能非常高。

7.2 通过编程智能体安装并运行智能体

最简单的方法是让 Codex 或 Claude Code 为你从源代码下载、安装并运行这些系统。以下是一些提示词模板。

7.2.1. 安装 *Rethlas*。

```
Help me download Rethlas from https://github.com/frenzymath/Rethlas.  Check that it
runs correctly on my laptop and that its MCP tools work.
```

7.2.2. 运行 *Rethlas*。

```
Help me run Rethlas, with GPT-5.6 Sol at max effort as the generation agent and
GPT-5.6 Sol at xhigh effort as the verification agent, on the following problem:
prove or disprove the BSD conjecture.
```

7.2.3. *macOS* 上的 *Pharos*。 默认情况下，Danus 和 Pharos 只能在 Linux 上运行。你可以在服务器上运行它们（例如亚马逊云科技，或斯坦福大学的 FarmShare；大多数服务器运行 Linux），也可以让你的编程智能体将它们适配到 macOS：

```
Help me install Pharos from https://github.com/frenzymath/Pharos and adapt it to
macOS. Check that it runs correctly on my laptop and that its MCP tools work.
```

7.2.4. 桌面应用：*Rethlas-OSV*。 Rethlas-OSV 是一款适用于 macOS 和 Linux 的桌面应用。它使用 OSVerify 运行 Rethlas 以进行问题求解，并单独使用 OSVerify 审阅论文。请从 https://github.com/OSVerify/Rethlas-OSV/releases 下载；分步指南位于 https://srliu.com/blog/rethlas-osv-guide/。首次尝试时，我们建议使用一到三个并行分支和 20 个轮次；之后你可以通过增加轮次来继续运行。

7.3 手动运行和自定义智能体

更详细的指导请见 https://srliu.com/blog。

7.3.1. 准备工作。 你需要 Codex CLI（参见 https://learn.chatgpt.com/docs/codex/cli#getting-started），并且熟悉终端（参见 https://srliu.com/blog/terminal-guide/）。

7.3.2. *Rethlas*。 验证器以本地网络服务的形式运行，生成器则是一个会调用它的 Codex 智能体。使用 tmux 会更方便，但并非必需。

安装：

```
git clone https://github.com/frenzymath/Rethlas.git && cd Rethlas

# 1. set up the verification agent
cd agents/verification && python3 -m venv .venv && source .venv/bin/activate
pip install -r requirements.txt && deactivate && cd ../..

# 2. set up the generation agent
```

```
cd agents/generation && python3 -m venv .venv && source .venv/bin/activate
pip install -r mcp/requirements.txt
```

用法:

```
# 1. start the verification service
cd Rethlas/agents/verification && source .venv/bin/activate
uvicorn api.server:app --host 0.0.0.0 --port 8091

# 2. in another shell: run the generation agent on a problem
cd Rethlas/agents/generation && source .venv/bin/activate
PROBLEM_FILE=data/my_problem.md ./tests/run_example.sh
```

将每个问题写入 Markdown 文件，并将 `PROBLEM_FILE`设置为该文件的路径。环境变量 `MODEL`、`REASONING_EFFORT` 和 `VERIFY_URL`分别用于选择模型、工作量级别和验证器。可将参考资料放置在 `data/<problem>.refs/`中；相比 PDF，优先使用 Markdown 和 LaTeX 文件。

输出包括 `logs/<id>/`中的日志、 `memory/<id>/`中的笔记本，以及 `results/<id>/blueprint.md`中的蓝图；蓝图通过验证后会被重命名为 `blueprint_verified.md` 。

7.3.3. *Danus*。 克隆代码仓库 https://github.com/frenzymath/Danus，然后从其根目录运行以下命令。

```
# 1. install the toolchain (Node, Python venv, Codex CLI) into runtime/
bash scripts/bootstrap.sh
# 2. copy the config templates and fill in YOUR keys (never committed)
cp config/danus.env.example config/danus.env
cp config/codex.env.example config/codex.env   # OpenAI-compatible endpoint
# 3. health check, then start the verify service (required for proving)
bash scripts/doctor.shbash scripts/services.sh up verify
# 4. start Codex at the repository root; the first run calls ‘initialize‘.
#    The flag below lets the main agent act without approval prompts, with
#    your shell privileges: run Danus on an isolated, disposable host, and
#    read docs/security-and-trust.md first.
codex --dangerously-bypass-approvals-and-sandbox
```

7.3.4. *OSVerify*。 克隆代码仓库 https://github.com/OSVerify/OSVerify。你需要 Python 3。12+， uv、经过身份验证的 CodexCLI，以及用于生成 PDF 报告的 XeLaTeX 或 LuaLaTeX。

```
uv syncuv run osverifier demo --run-dir runs/demo     # offline demo,
 no model calls# review a solution
uv run osverifier verify --question question.md --solution solution.md \
```

```
    --request-id problem-001 --model MODEL_ID --decompose \
--run-dir runs/problem-001# review a paper
uv run osverifier verify --review-type paper --decompose --question scope.md \
--solution paper.tex --request-id paper-v1 --model MODEL_ID \
--run-dir runs/paper-v1# solve with Rethlas, using OSVerify as verifier
uv sync --extra rethlas
uv run osverifier-rethlas --problem problem.md --run-dir runs/my-problem \
--generation-branches 4 --max-rounds 10
uv run osverifier-gui                              # local GUI
uv run osverifier export --run-dir runs/problem-001 --format latex \
--output report.tex
```

7.4 Pharos 及其操作代理

Pharos 旨在提供交互式体验，并通过编码智能体的 CL 进行管理 I.

7.4.1. 要求。 Pharos 可在 Linux 上运行，需要 Python 3.10+、Git、tmux、Chromium（用于报告）以及 `pdflatex`或 `tectonic` （用于论文）。它也可以轻松适配 macOS。

7.4.2. 启动操作代理。

```
git clone https://github.com/frenzymath/Pharos.git && cd Pharos
codex --yolo                          # from the repository root
```

现在你正在与操作代理对话。请它设置 Pharos，并确认它能在你的机器上正常运行。然后用自然语言描述你的问题并说明需求；操作代理会为你创建并启动项目。研究开始前，系统会请你评审问题陈述和初始指导。之后，你可以直接向项目的主代理发送指导，也可以通过操作代理发送。

若要自定义 Pharos，请向操作代理描述你的需求，并让它实施更改。

7.5 良好实践

7.5.1. 准确陈述问题。避免滥用记号。如果该陈述可能为假，请说“证明或证伪”。

7.5.2. 提供参考资料。你可能需要自行下载受版权保护的书籍。在可能的情况下，请提供 LaTeX 或 Markdown 文件，而不是 PDF 文件。

7.5.3. 选择合适的工具。 参见第 7.1节。

7.5.4. 保持透明。 在论文中披露人工智能系统的使用情况，说明哪些部分由人工智能生成，并对内容的正确性负责。

7.6 如何贡献

工具框架设计是数学专业知识能够直接发挥作用的领域。我们非常欢迎各类贡献；请与我们联系。

7.6.1. 找出失效模式。 在你理解的问题上运行一个系统，找出它如何以及为何失效，并报告结果。

7.6.2. 寻找新的应用。 探索人工智能为数学作出有意义贡献的新方式：解决新领域的问题、构建理论、改进写作或理解等。例如， [Bos26] 是首个使用 Pharos 开发框架、将两个问题联系起来的实例。

7.6.3. 构建数据集和基准测试。 这些对于评估智能体系统和训练模型都很重要。

7.6.4. 将工作流程提炼为技能和编排。 人类的经验为工具框架设计提供直觉。根据你自己的研究经验改进工具框架。

8 未来方向及在其他领域的应用 s

8.1 开放方向

8.1.1. 随着模型进步，简化并改进工具框架。 随着模型变得更强大，用来弥补其弱点的组件将不再必要（第 4 节）。哪些组件应该移除？更强大的模型又能支持哪些更具雄心的工具框架？

8.1.2. 从验证到理解与写作。 人工智能如何帮助人类理解论文，并协助撰写更清晰的论文？这是我们正在进行的研究课题。

8.1.3. 人机协作。 我们需要更有效的方式，让专家能够引导长时间运行的任务（如 Danus 和 Pharos），也需要提升智能体的写作与交流技能。

8.1.4. 创造力与理论构建。 当前系统的工作大多局限于“已知技术的凸包”（这并不是一个定义明确的概念！）。我们如何才能让它们更具创造力，甚至能够构建新理论？

8.1.5. 评估。 如何严谨地对研究级智能体系统进行基准测试？目前，智能体系统的评估仍主要依赖实证方法。

8.1.6. 自我改进。 工具框架能否递归地改进自身？

术语表

蓝图
Rethlas 的证明文档：由一系列包含陈述和证明的引理组成，并以主定理收尾。

压缩
上下文窗口填满时，用摘要替换旧的上下文。

上下文窗口模型一次能够关注的文本。

事实、事实图
经验证器认可的陈述及其证明；事实及其依赖关系构成的有向无环图（Danus、Pharos）。

假阴性率、真阴性率
假阴性率（有效解答被拒绝的比例）和真阴性率（无效解答被拒绝的比例），其中有效解答为正类。

工具框架
智能体中除模型权重以外的一切。

MCP模型上下文协议，一种向智能体公开工具的标准方式。

推理力度
A 控制模型每一步思考时长的设置（`high`、 `xhigh`、 `max`，.....） p.

路由板Pharos 对各种方法、其状态、证据和障碍的记录。

技能针对重复性任务的 Markdown 指令文件，智能体会在相关时加载。

子智能体由另一个智能体启动、拥有独立上下文的智能体。

测试时计算解题过程中花费的计算资源，而非训练过程中花费的计算资源。

参考文献

[Abo+26] M. Abouzaid、A. J. Blumberg、M. Hairer、J. Kileel、T. G. Kolda、P. D. Nelson、D. Spielman、N. Srivastava、R. Ward、S. Weinberger 和 L. Williams。First Proof。2026年。arXiv：2602.05192。url: https://1stproof.org/。[Ach+25] T. Achim、A. Best、A. Bietti、K. Der、M. Fédérico、S. Gukov、D. Halpern-Leistner、K. Henningsgard、Y. Kudryashov、A. Meiburg、M. Michelsen、R. Patterson、E. Rodriguez、L. Scharff、V. Shanker、V. Sicca、H. Sowrirajan、A. Swope、M. Tamas、V. Tenev、J. Thomm、H. Williams 和 L. Wu。Aristotle：国际数学奥林匹克级别的自动定理证明。2025年。arXiv：2510.01346。[Alo+26] N. Alon、T. F. Bloom、W. T. Gowers、D. Litt、W. Sawin、A. Shankar、J. Tsimerman、V. Wang 和 M. M. Wood。关于单位距离猜想被否证的评论。2026年。arXiv：2605.20695。[Alp26] L. Alpöge。宣布黎曼猜想证明的未命名笔记。2026年11月20日。url： https://alpo.ge/remark.pdf。[AVR26] L. Alpöge、P. Voinov 和 S. Reynolds-Haertle。2000阶以下的 Hadamard 矩阵。在 X 上发布的公告。2026年8月12日。url： https://x.com/__alpoge__/status/2087504785952182273。[Ame26] 美国数学学会。数学与人工智能。2026年。url： https://www.ams.org/ai/ai。[AH77] K. Appel 和 W. Haken。“每个平面图都可以进行四色着色。第一部分：放电法”。载于：《伊利诺伊数学杂志》21.3（1977），第429–490页。[Ass26] 人类数学协会。人类数学协会（AHM）。2026年。url: https://www.ahmath.org/。[BBWAM26] S. Barkallah、L. Bailey、K. Wen、M. Abouzaid 和 T. Ma。用于自动证明验证的伪形式化。2026年。arXiv：2605.20531。[BCH26] A. Bastounis、F. Circelli 和 A. C. Hansen。纳维–斯托克斯方程在翻译中迷失：为何 Lean 对人工智能自动形式化的验证并不能保证自然语言证明的正确性。2026年。arXiv：2610.08144。[Bos26] H. Bosch。关于 ε–正则性的一个部分定理： $\mathbb{R}^n$中的积分平稳测地网络。2026年。arXiv：2607.23872。[BA26] T. Buckmaster 和 L. Alpöge。纳维–斯托克斯方程的有限时间爆破。2026年11月20日。url: https://github.com/tristanbuckmaster/navier-stokes。[CFFG14] P.-J. Cahen、M. Fontana、S. Frisch 和 S. Glaz。“交换环理论中的开放问题”。载于：《交换代数：交换环、整值多项式和多项式函数的最新进展》。Springer，2014年，第353–375页。[CEWW24] F. Charton、J. S. Ellenberg、A. Z. Wagner 和 G. Williamson。PatternBoost：在人工智能的少许帮助下进行数学构造。2024年。arXiv：2411.00566。[Che+25] J. Chen、W. Chen、J. Du、J. Hu、Z. Jiang、A. Jie、X. Jin、X. Jin、C. Li、W. Shi、Z. Wang、M. Wang、C. Wei、S. Wei、H. Xin、F. Yang、W. Gao、Z. Yuan、T. Zhan、Z. Zheng、T. Zhou 和 T. H. Zhu。Seed-Prover 1.5：通过从经验中学习掌握本科水平的定理证明。2025年。arXiv：2512.17260。

[Che26] R. Cheng。拟阵构造集的切丛类。2026年。arXiv：2606.22650[math.AG]。网址：https://arxiv.org/abs/2606.22650。[CL26] R. Cheng 和 S. Liu。拟阵的 Kazhdan–Lusztig 多项式不一定是单峰的。2026年。arXiv：2607.24186。[CLG26] R. Cheng、S. Liu 和 G. Gao。拟阵的切丛类与奇妙紧化。2026年。arXiv：2607.05835。[CLN26] Z. Chi、S. Liu 和 P. Nie。关于 Manin 常数。2026年。arXiv：2610.25672。[CGP15] B. Conrad、O. Gabber 和 G. Prasad。《伪约化群》。第 2 版。剑桥大学出版社，2015。[CP17] B. Conrad 和 G. Prasad。“伪约化群的结构与分类”。载于：《代数群：结构与作用》。第 94 卷。《纯数学研讨会论文集》。美国数学会，2017，第 127–276 页。[DAR25] DARPA。expMath：数学指数化。https://www.darpa.mil/research/programs/expmath-exponential-mathematics。项目描述。2025年。[Dav+21] A. Davies、P. Veličković、L. Buesing、S. Blackwell、D. Zheng、N. Tomašev、R. Tanburn、P. Battaglia、C. Blundell、A. Juhász、M. Lackenby、G. Williamson、D. Hassabis 和 P. Kohli。“通过人工智能引导人类直觉来推进数学”。载于：《自然》600.7887（2021），第 70–74 页。doi：10.1038/s41586-021-04086-x。[Don+26] B. Dong、G. Gao、B. Guo、Z. Sun、B. Wu 和 S.-Y. Xie。通过射影簇上的喷射构造具有非 nef 正则丛的有理曲线。2026年。arXiv：2609 . 28465。附录 B 描述了 Pharos；代码位于 https://github.com/frenzymath/Pharos。[DJ26] X. Dou 和 Z. Jin。Sobolev 圆映射的次数不等式中的退化常数：关于 Brezis 提出的若干问题。2026年。arXiv：2605.24626。[DLTTM24] Y. Du、S. Li、A. Torralba、J. B. Tenenbaum 和 I. Mordatch。“通过多智能体辩论提升语言模型的事实性与推理能力”。载于：《第 41 届国际机器学习会议论文集》（ICML 2024）。第 235 卷。《机器学习研究论文集》。2024，第 11733–11763 页。[EKLÓ26] R. Egan、I. Kotsireas、S. London 和 P. Ó Catháin。Hadamard 矩阵构造的最新进展。2026年。arXiv：2610.27175。[Eng26] P. Engel。S^6上的复结构。2026年。arXiv：2609.38442。[Fen26] T. Feng。算术 Hirzebruch 比例定理的特征权。2026年。arXiv：2601.23245 [math.RT]。网址：https://arxiv.org/abs/2601.23245。[Fen+26a] T. Feng、J. Jung、S.-h. Kim、C. Pagano、S. Gukov、C.-C. Tsai、D. Woodruff、A. Javanmard、A. Mokhtari、D. Hwang、Y. Chervonyi、J. N. Lee、G. Bingham、T. H. Trinh、V. Mirrokni、Q. V. Le 和 T. Luong。Aletheia 自主攻克 FirstProof。2026年。arXiv：2602.21201。[Fen+26b] T. Feng、T. H. Trinh、G. Bingham、D. Hwang、Y. Chervonyi、J. Jung、J. Lee、C. Pagano、S.-h. Kim、F. Pasqualotto、S. Gukov、J. N. Lee、J. Kim、K. Hou、G. Ghiasi、Y. Tay、Y. Li、C. Kuang、Y. Liu、H. Lin、E. Z. Liu、N. Nayakanti、X. Yang、H.-T. Cheng、D. Hassabis、K. Kavukcuoglu、Q. V. Le 和 T. Luong。迈向自主数学研究。2026年。arXiv：2602.10177。

[Gao26] G. Gao。工具框架仍然重要吗？斯坦福大学演讲幻灯片。2026年 11 月。[GJJQD24] G. Gao、H. Ju、J. Jiang、Z. Qin 和 B. Dong。“Mathlib4 的语义搜索引擎”。载于：《计算语言学协会论文集：EMNLP 2024》。2024，第 8001–8013 页。[Gao+26] G. Gao、Z. Sun、B. Wu、S. Liu、J. Jiang 和 B. Dong。Pharos：长时程研究智能体。准备中。G. Gao、Z. Sun、B. Wu 和 S. Liu 作出同等贡献。2026年。[Gao+24] G. Gao、Y. Wang、J. Jiang、Q. Gao、Z. Qin、T. Xu 和 B. Dong。Herald：自然语言标注的 Lean 4 数据集。2024年。arXiv：2410.10878。[Gut26] L. Guth。《数学对我意味着什么》。2026年。arXiv：2609.32028 [math.HO]。网址：https://arxiv.org/abs/2609.32028。[Hub+25] T. Hubert、R. Mehta、L. Sartran、M. Z. Horváth、G. Žužić、E. Wieser、A. Huang、J. Schrittwieser、Y. Schroecker、H. Masoom、O. Bertolli、T. Zahavy、A. Mandhane、J. Yung、I. Beloshapka、B. Ibarz、V. Veeriah、L. Yu、O. Nash、P. Lezeau、S. Mercuri、C. Sönne、B. Mehta、A. Davies、D. Zheng、F. Pedregosa、Y. Li、I. von Glehn、M. Rowland、S. Albanie、A. Velingker、A. Schmitt、S. Lockhart、E. Hughes、H. Michalewski、N. Sonnerat、D. Hassabis、P. Kohli 和 D. Silver。“通过强化学习进行奥林匹克级形式化数学推理”。载于：《自然》651.8106（2025），第 607–613 页。doi：10.1038/s41586-025-09833-y。[Irv+16] G. Irving、C. Szegedy、A. A. Alemi、N. Eén、F. Chollet 和 J. Urban。“DeepMath——用于前提选择的深度序列模型”。载于：《神经信息处理系统进展》（NeurIPS 2016）。2016。[Jia+26] J. Jiang、Y. Li、Z. Sun、Y. Wang、L. Xiao 和 J. Yu。关于由 Rethlas 解决的交换代数中的若干公开问题。2026 年。arXiv：2605.25259。[Ju+26] H. Ju、G. Gao、J. Jiang、B. Wu、Z. Sun、S. Liu、L. Chen、Y. Wang、Y. Wang、Z. Wang、W. He、P. Wu、L. Xiao、R. Liu、B. Dai 和 B. Dong。通过形式化验证实现自动猜想解决。2026年。arXiv：2604.03789。[Liu14] C. Liu。《三体》。Ken Liu 译。纽约：Tor Books，2014。[Liu15] C. Liu。《黑暗森林》。Joel Martinsen 译。纽约：Tor Books，2015。[Liu+26] J. Liu、G. Gao、Z. Sun、B. Wu、S. Liu、J. Jiang、H. Ju、L. Chen、R. Cheng、X. Zhang 和 B. Dong。Danus：利用事实图谱记忆编排数学推理智能体。2026年。arXiv：2607.06447。技术报告：https://frenzymath.com/blog/danus/。[Liu26a] S. Liu。相对几何朗兰兹中的自守交换子关系。2026年。arXiv：2610.05781。[Liu26b] S. Liu。关于 Plancherel 代数的性质。Hexagon 预印本 hexagon:2609.00128v1。2026年。网址：https://hexagonmath.org/2609.00128v1。[Mat26] MathOverflow。在重大数学发展中使用人工智能，尤其是大语言模型的示例。2026年。网址：https://mathoverflow.net/questions/502120/。[McC97] W. McCune。“Robbins问题的解”。载于：《自动推理杂志》19.3（1997），第 263–276 页。

[MKADR15] L. de Moura、S. Kong、J. Avigad、F. van Doorn 和 J. von Raumer。《Lean 定理证明器》。载于：《自动演绎——CADE-25》，第9195卷，《计算机科学讲义》。2015年，第378–388页。[Nie+26] F. Nie、H. Ye、S. Liu、Z. Wang、S. Ermon 和 J. Zou。在有限专家评审下的单侧验证。2026年。arXiv：2610.18089。[Nov+25] A. Novikov、N. V˜u、M. Eisenberger、E. Dupont、P.-S. Huang、A. Z. Wagner、S. Shirobokov、B. Kozlovskii、F. J. R. Ruiz、A. Mehrabian、M. P. Kumar、A. See、S. Chaudhuri、G. Holland、A. Davies、S. Nowozin、P. Kohli 和 M. Balog。AlphaEvolve：用于科学与算法发现的编程智能体。2025年。arXiv：2506.13131。[Ope26a] OpenAI。受迫纳维–斯托克斯方程的有限时爆破。2026年9月8日。网址：https://openai.com/index/navier-stokes-solution/。Lean证明证书见 https://github.com/openai/NavierStokesAndEuler。[Ope26b] OpenAI。First Proof? 技术报告。OpenAI，2026年2月20日。网址：https://cdn.openai.com/pdf/26177a73-3b75-4828-8c91-e8f1cf27aaa0/oai_first_proof.pdf。[Ope26c] OpenAI。模型反驳离散几何猜想。2026年5月。网址：https://openai.com/index/model-disproves-discrete-geometry-conjecture/。[Ope26d] OpenAI。分享数学领域的人工智能进展。2026年10月6日。网址：https://openai.com/index/sharing-ai-progress-in-mathematics/。手稿见 https://github.com/openai/math。[Ope26e] OpenAI。数学与理论计算机科学的十项进展。2026年8月1日。网址：https://openai.com/index/ten-advances-in-mathematics/。[PY26] X. Pan 和 J. Yu。曲线的 p-进Simpson对应中的提升独立性问题。2026年。arXiv：2605.29947。[Pan+26] S. Pandit、A. Xu、X.-P. Nguyen、Y. Ming、C. Xiong 和 S. Joty。“Hard2Verify：开放式前沿数学的逐步验证基准”。载于：计算语言学协会年会论文集（ACL 2026）。2026年，第22502–22517页。arXiv：2510.13744。[PS20] S. Polu 和 I. Sutskever。用于自动定理证明的生成式语言建模。2020年。arXiv：2009.03393。[26] Proofs and Prompts：来自公共休息室的笔记。2026年。网址：https://proofsandprompts.com/。[RHQ26] T. M. F. Ramos、R. Hulak 和 R. J. G. B. de Queiroz。对雅可比猜想的一个显式反例进行形式化验证。《形式化证明档案库》。2026年7月20日。网址：https://isa-afp.org/entries/Jacobian_Counterexample.html。[Ras25] S. Raschka。从头实现字节对编码（BPE）分词器。访问日期：2026年11月17日。2025年1月。网址：https://sebastianraschka.com/blog/2025/bpe-from-scratch.html。

[Ret26] Retraction Watch。OpenAI从其发布的722篇未解决数学问题手稿中撤回预印本。2026年10月8日。网址：https://retractionwatch.com/2026/10/08/openai-withdraws-preprints-722-manuscripts-unsolved-math-problems/。[Sch22] P. Scholze。“Liquid Tensor Experiment”。载于：Experimental Mathematics 31.2（2022年），第349–354页。[Sci26] 《科学美国人》。Anthropic数学家声称证明了黎曼猜想。2026年11月20日。网址：https://www.scientificamerican.com/article/anthropic-mathematician-claims-to-prove-riemann-hypothesis-most-famous-problem-in-all-of-math/。[Sev26] 多位投稿者。今年的招聘：委员会的建议与变化。Proofs and Prompts。2026年11月14日。网址：https://proofsandprompts.com/2026/11/14/hiring-this-year-advice-and-changes-from-committees/。[SB26] A. Shelley 和 H. Bosch。碰撞系统及其在测地线网络和点弹子台中的应用。Hexagon预印本 hexagon:2610.00601v1。2026年10月29日。网址：https://hexagonmath.org/2610.00601v1。[Tan26] Q. Tang。Mirković–Vilonen基与theta基的相等性。Hexagon预印本 hexagon:2611.00006v1。2026年。网址：https://hexagonmath.org/2611.00006v1。[Tao26] T. Tao。对雅可比猜想反例的解读。《What's new》（博客）。2026年7月21日。网址：https://terrytao.wordpress.com/2026/07/21/a-digestion-of-the-jacobian-conjecture-counterexample/。[Tao+26] T. Tao等。人工智能对Erdős问题的贡献。GitHub wiki，teorth/erdosproblems。2026年。网址：https://github.com/teorth/erdosproblems/wiki/AI-contributions-to-Erd%C5%91s-problems。[The20] mathlib社区。“Lean数学库”。载于：第9届ACM SIGPLAN国际认证程序与证明会议论文集（CPP 2020）。2020年，第367–381页。[Thu94] W. P. Thurston。《数学中的证明与进展》。1994年。arXiv：math/9404236 [math.HO]。网址：https://arxiv.org/abs/math/9404236。[TWLHL24] T. H. Trinh、Y. Wu、Q. V. Le、H. He 和 T. Luong。“无需人类示范解决奥林匹克几何问题”。载于：《自然》625（2024年），第476–482页。[Tri26] V. Trivedy。智能体工具框架的剖析。LangChain博客。2026年3月10日。网址：https://langchain.com/blog/the-anatomy-of-an-agent-harness。[TMCCM24] G. Tyen、H. Mansoor、V. Cărbune、P. Chen 和 T. Mak。“大语言模型无法发现推理错误，但在给出错误位置后可以纠正错误”。载于：计算语言学协会论文集：ACL 2024。2024年，第13894–13908页。doi：10.18653/v1/2024.findings-acl.826。[Wag21] A. Z. Wagner。通过神经网络进行组合学构造。2021年。arXiv：2104.14516。[Wan+25] H. Wang、R. Xie、Y. Wang、G. Gao、X. Yu 和 B. Dong。Aria：通过依赖图进行检索和迭代式自动形式化的智能体。2025年。arXiv：2510.04520。

[Wei25] A. Wei。 *OpenAI* 的实验性推理模型在 *2025*年国际数学奥林匹克 中取得金牌级表现。发布于 X；由 S. Willison 总结。2025年7月19日。网址：https://simonwillison.net/2025/Jul/19/openai-gold-medal-math-olympiad/。[Wei26] M. Weinreich。 人工智能生成的数学危机。即将刊载于《美国数学学会通告》。2026年。arXiv：2608.02859。[Won24] M. Wong。 我们正在进入数学的未知领域。《大西洋月刊》。与陶哲轩的访谈。2024年10月。[Yan+23] K. Yang、A. M.Swope、A. Gu、R. Chalamala、P. Song、S. Yu、S. Godil、R. Prenger 和 A. Anandkumar。“LeanDojo：利用检索增强语言模型进行定理证明”。载于：《神经信息处理系统进展》（*NeurIPS 2023*）。2023年。arXiv：2306.15626。[Zhu13] 庄子。《庄子全书》。Burton Watson 译；参见第一章《逍遥游》。纽约：哥伦比亚大学出版社，2013年。