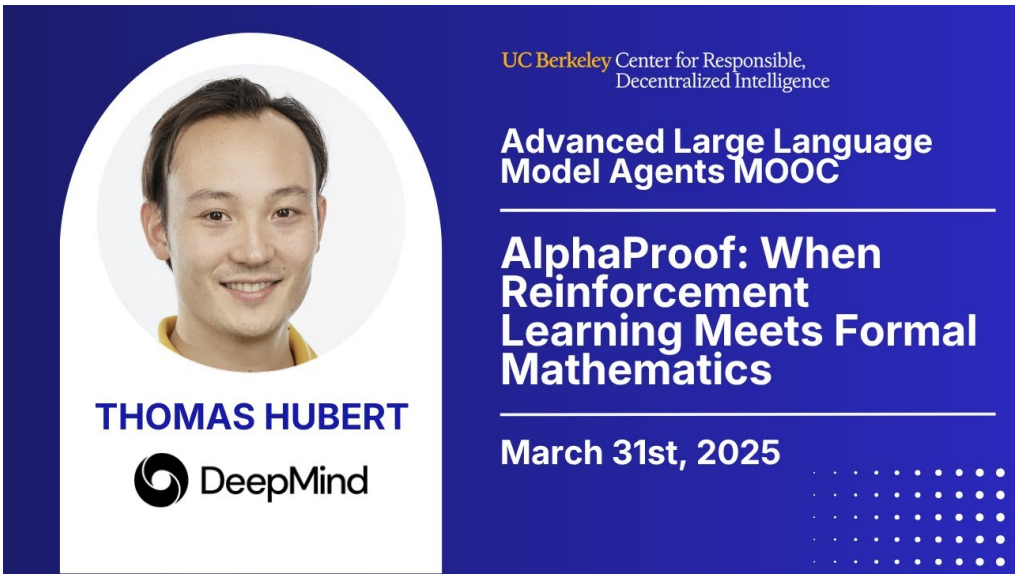
Berkeley CS294/194-280 Advanced Large Language Model Agents, Spring 2025

AlphaProof：当强化学习遇到形式数学

cnfjlhj & Codex

2026 年 5 月 28 日



讲座：AlphaProof: when reinforcement learning meets formal mathematics
主讲：Thomas Hubert, Google DeepMind
视频作者/频道：Berkeley RDI
发布日期：2025-03-31
视频时长：1:14:08
视频链接：https://www.youtube.com/watch?v=3gaEMscOMAU
课程页与 slides：课程主页；官方 slides

目录

1 这节课的核心问题

这节课的主题不是“用大模型做数学题”的普通综述，而是一个更窄、更硬的命题：如果数学可以被放进一个形式系统里，Lean 可以给出可机检的反馈，那么强化学习能不能像在围棋、游戏和矩阵乘法里一样，通过大规模试错发现新的证明路径？AlphaProof 的答案是：可以，但前提不是“大语言模型更会聊天”，而是把数学重构成一个可以搜索、验证、训练、再搜索的环境。

Thomas Hubert 的讲座有两条线并行推进。第一条是数学与形式化：为什么从自然语言证明走向符号和机器检查，是数学长期演化的一部分。第二条是强化学习：为什么 AlphaZero 系列的经验可以迁移到形式数学，但不能简单照搬。两条线交汇的地方，就是 AlphaProof。

一页压缩版

AlphaProof 把 Lean 看成数学环境，把证明步骤看成动作，把 Lean 的检查结果看成可靠反馈。它不是直接让模型“写一篇像人的证明”，而是让模型在形式化问题周围生成、搜索、验证和强化训练，最后在 IMO 2024 中解决了 P1、P2、P6；几何题 P4 由 AlphaGeometry 解决。

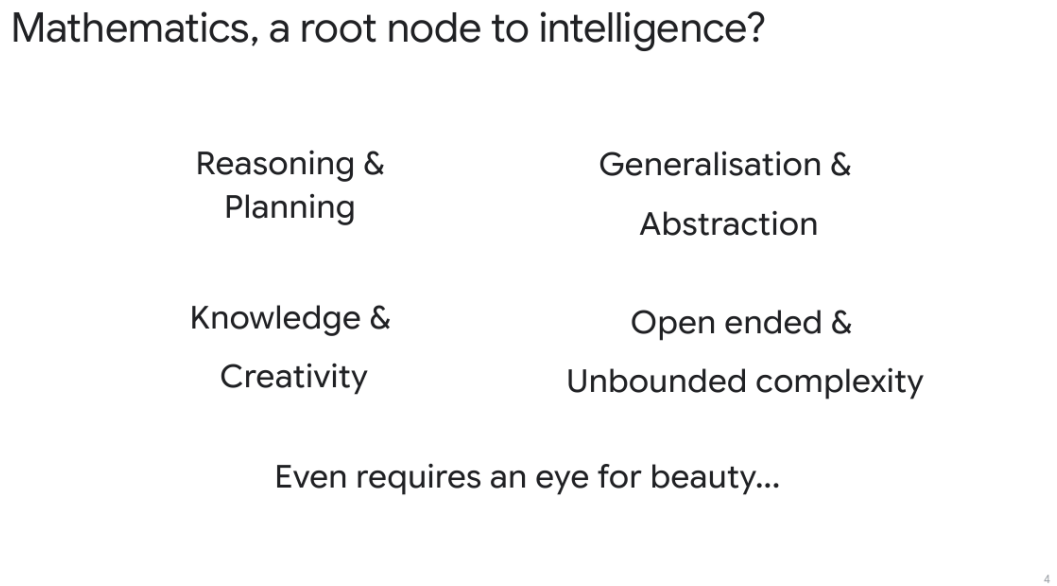


图 1: 讲座开场把数学放在推理、规划、泛化、抽象、知识与创造力的交汇点。图源：官方 slides p.4。

这份笔记会按“动机、环境、强化学习配方、IMO 实战、方法结构、局限与下一步”的顺序重建讲座，而不是逐页翻译 slides。这样更适合后续衔接自动形式化和自动定理证明的工作。

1.1 本章小结

AlphaProof 的主线可以写成一个三段式：数学需要可信验证；Lean 提供可交互、可检查的数学环境；强化学习和搜索把这个环境变成可扩展训练场。理解这一点，后面的形式化、RL 和 test-time RL 才不会散成几个孤立技术词。

2 为什么先谈形式数学

Hubert 从数学史讲起，是为了说明形式化不是突发的工程癖好，而是数学本身一直在追求的方向。古希腊以来，证明的地位越来越核心；从文字算法到符号代数，数学表达越来越压缩、可传递、可复用。形式化进一步把证明放进机器可检查的语言中，让“这一步是否成立”不再只依赖人的读后判断。

在讲座里，形式化被赋予了几种价值：

- **严谨与清晰**：每个对象、假设和推理步骤必须被写清楚。
- **效率与沟通**：机器可读的数学可以被搜索、复用、重构和协作维护。
- **抽象与泛化**：统一的定义和库让不同分支之间更容易共享结构。
- **信任**：检查证明这件事可以交给计算机，人的精力可以转向发现和组织。

“形式化”不是把数学降级成代码

形式化不是把数学变成机械抄写。它真正改变的是反馈机制：自然语言证明通常要靠专家读懂并判断，而 Lean 这样的 proof assistant 会在每个状态上告诉你目标、上下文、缺口和错误。这使证明可以被程序化搜索。

2.1 Lean 与 Mathlib：环境比模型更基础

Lean 在讲座中有三重身份：编程语言、定理证明器、交互式证明助手。它的重要性不只是“能写证明”，而是提供一个数学环境。用户写 theorem statement 和 proof，Lean 返回局部上下文、当前目标和检查结果。对于 RL 系统来说，这正是环境反馈。

Mathlib 则是 Lean 的大型数学库。Hubert 强调 Mathlib 覆盖了大量本科以及本科以上数学，但覆盖不均匀，还有明显空洞，尤其是 IMO 需要的二维欧几里得几何等部分。这一点后面会直接影响 AlphaProof 的 IMO 参赛策略。

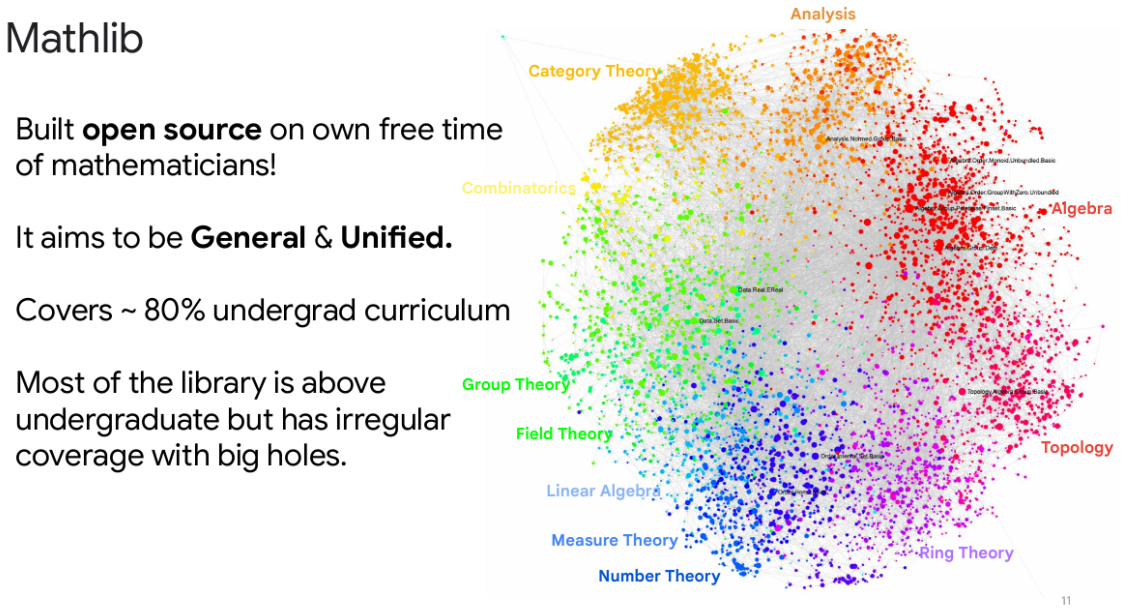


图 2: Mathlib 覆盖分析、代数、组合、拓扑、数论等大量领域，但覆盖仍然不均匀。图源：官方 slides p.11。

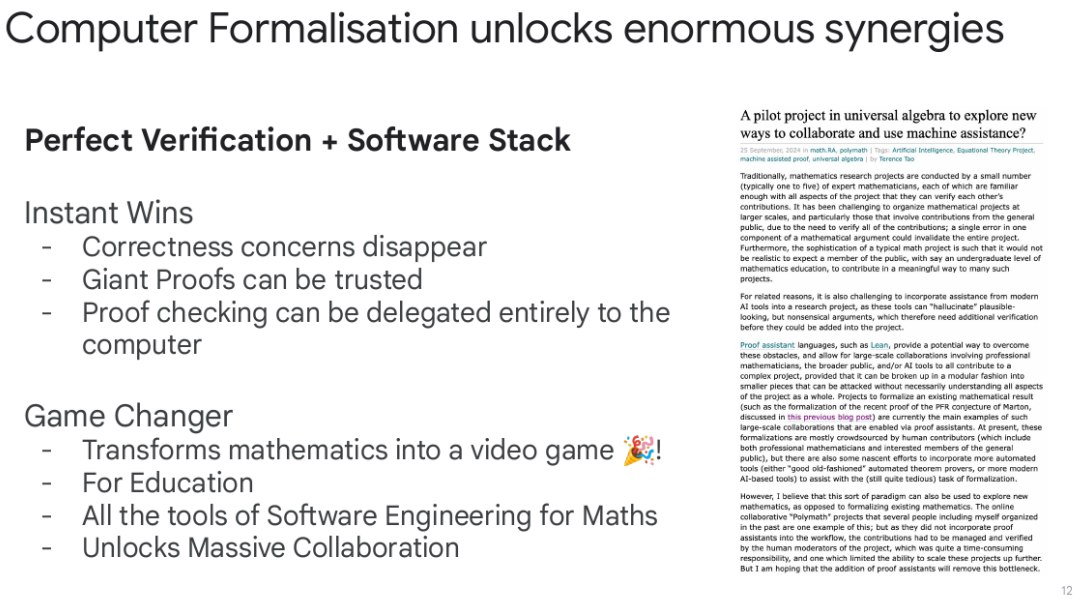


图 3: 形式化数学带来的协同：完美验证、软件工程工具、教育与大规模协作。图源：官方 slides p.12。

2.2 采用率低说明了什么

讲座给出的现实约束很直接：采用 Lean 的数学家比例仍然很低，大约只有 0.1%–1%。原因包括学习曲线陡、时间投入大、工具和库仍在成熟中，以及许多数学家尚未感到研究上必须使用它。

这对 AlphaProof 很关键。它说明问题不是“Lean 已经自然成为数学主流，现在让 AI 加速一下”这么简单。更准确的说法是：AlphaProof 押注的是一个尚未完全普及但反馈极强的数学基础设施。这个基础设施一旦足够宽，AI 的搜索和学习能力才有真正落点。

不要把 Lean 当成普通 verifier

Lean 不是只在最后检查答案的裁判。对 AlphaProof 来说，它是训练环境、交互接口、搜索状态生成器和奖励来源。如果只把 Lean 理解成“最后验一下对不对”，就会误读整个方法。

2.3 本章小结

形式数学给 AlphaProof 提供了两个必要条件：第一，证明可以被拆成机器可检查的状态转移；第二，错误反馈足够密集，能支撑搜索和学习。没有 Lean/Mathlib 这样的环境，强化学习很难在数学证明中获得可靠试错信号。

3 从 AlphaZero 到数学证明

AlphaProof 的第二个根源是 DeepMind 的强化学习传统。Hubert 回顾了 DQN、AlphaGo、AlphaGoZero、AlphaZero、MuZero、AlphaTensor 等系统，核心不是为了罗列历史成就，而是为了抽象出一个“可迁移配方”：大规模试错、可靠反馈、强搜索、以及能从自我生成经验中改进的学习过程。

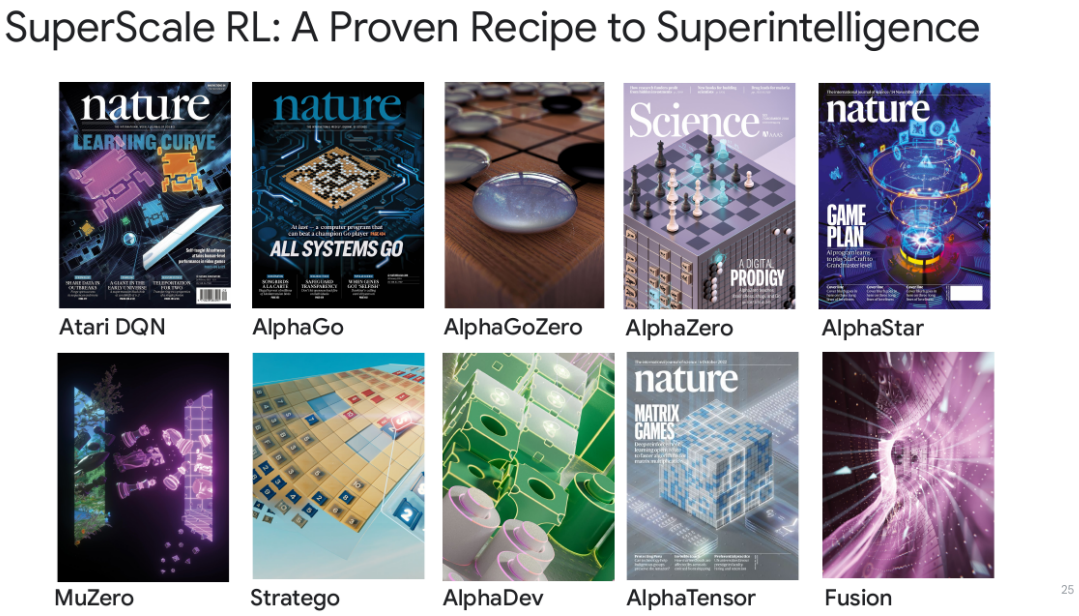


图 4: DeepMind 的强化学习谱系：从 Atari、围棋到 AlphaTensor。图源：官方 slides p.25。

3.1 Zero 哲学

讲座把 AlphaGoZero/AlphaZero 的哲学概括为：如果一个 agent 能够在环境中从空白状态出发，通过自我对弈或试错掌握任务，那么它就在实际发现策略和知识。这个思想诱人的地方是，它不必完全依赖人类示范数据。人类知识可以提供初始结构，但真正的提升来自环境内的探索和反馈。

迁移到数学时，这个哲学会遇到一个问题：围棋有明确规则和胜负，数学证明有没有类似的环境和反馈？AlphaProof 的回答是 Lean：proof state 就是状态，tactic 就是动作，Lean checker 返回的新状态、错误或完成证明就是反馈。

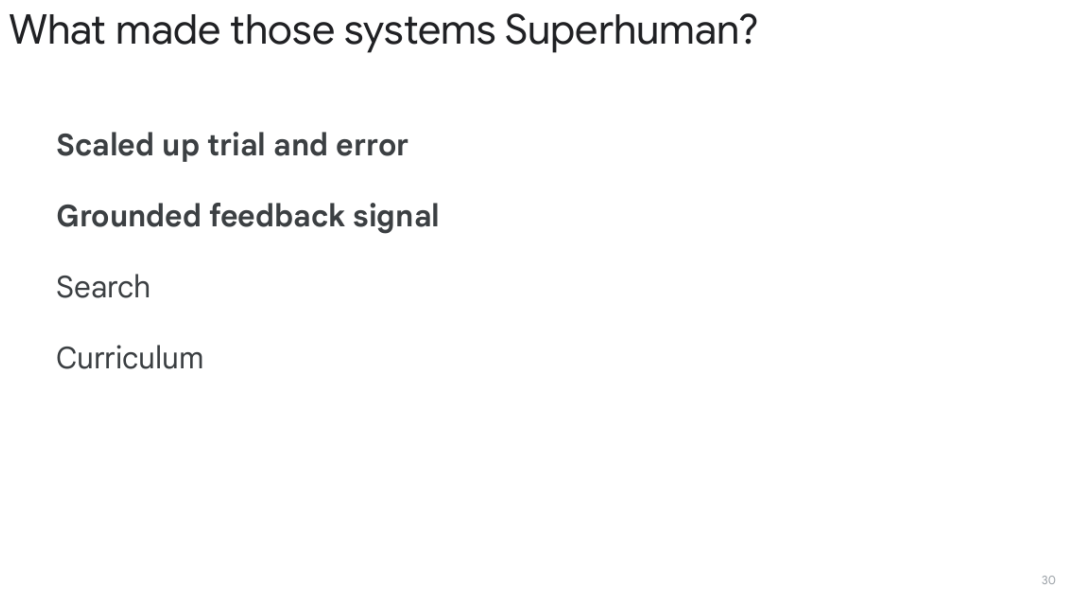


图 5: 走向超人系统的两个关键条件：大规模试错与 grounded feedback signal。图源：官方 slides p.30。

3.2 AlphaTensor 的启发

AlphaTensor 说明 AlphaZero 式方法不只适用于棋盘游戏。矩阵乘法算法搜索虽然不是游戏，却可以被形式化成一个环境：系统提出候选算法，环境评估其正确性和效率，搜索逐渐发现人类未必容易想到的结构。Hubert 用这条线把听众引向数学证明：证明同样可以看作一种行动序列，只要环境能判断行动是否有效。

可验证性是迁移的桥

RL 能否迁移到数学，不取决于数学是否“像游戏”，而取决于我们能不能构造出类似游戏的反馈结构。Lean 的价值正在这里：它让证明尝试从含糊的自然语言解释，变成可执行、可失败、可恢复、可搜索的状态转移。

3.3 本章小结

AlphaProof 借鉴的是 AlphaZero 的结构性经验，而不是表面任务形式。核心问题是：数学证明能否成为一个可探索环境? Lean 让这个问题有了工程答案，强化学习则提供了在环境中扩大搜索和改进策略的路线。

4 AlphaProof 的基础押注

AlphaProof 的 foundational bet 是：长期来看，完美验证会成为数学中最重要的属性之一。这里的“完美”不是说系统能自动发现所有证明，而是说一旦有形式证明，检查正确性可以极其可靠。这个押注把 AlphaProof 和只追求自然语言答案的数学 LLM 区分开来。

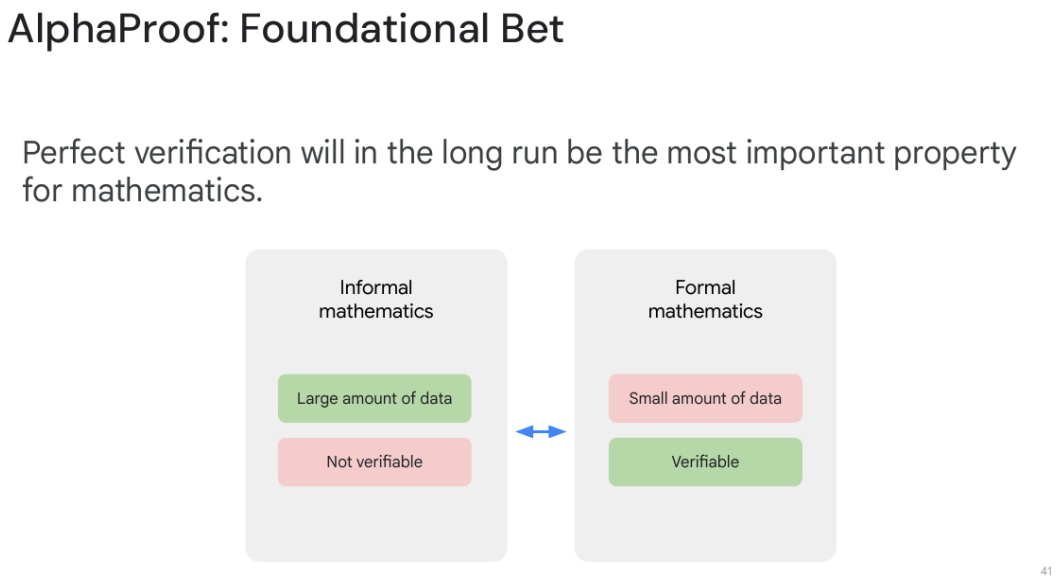


图 6: AlphaProof 的基础押注：完美验证将长期成为数学最重要的属性之一。图源：官方 slides p.41。

4.1 总计划：把 Lean 变成 RL 训练场

讲座中的 master plan 可以拆成两个方面。第一，Lean 允许系统在完全 in silico 的环境里探索数学：生成 proof state，尝试 tactic，接收 Lean 的检查。第二，LLM 提供语言与模式能力：它能从自然语言问题、已有证明库和中间状态中提出候选形式化与候选证明动作。

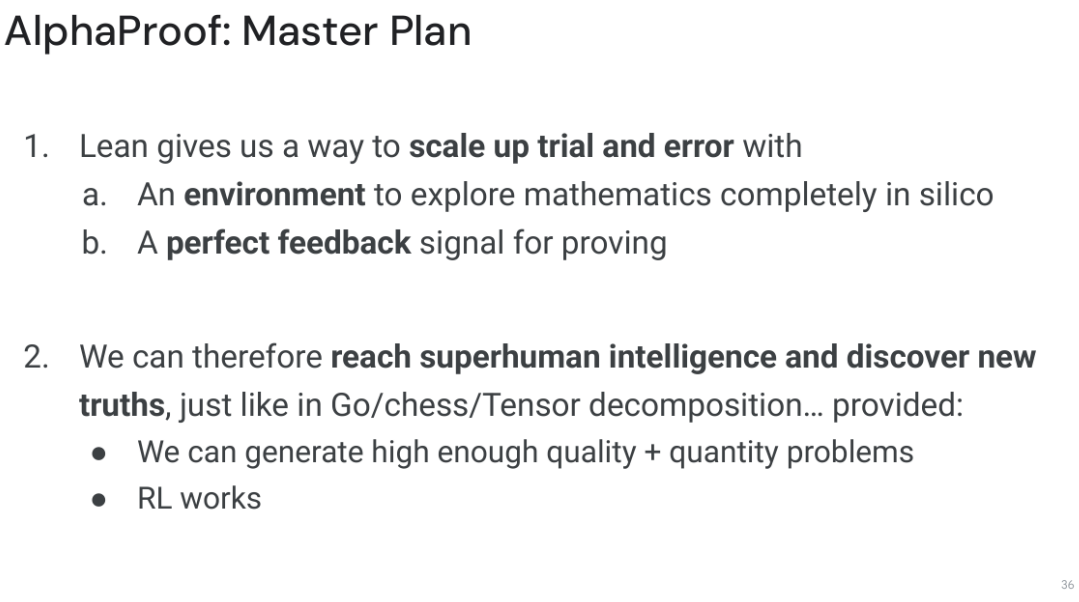


图 7: AlphaProof 总计划：Lean 提供环境与 grounded feedback，LLM/RL 提供探索和学习能力。图源：官方 slides p.36。

这也解释了为什么 AlphaProof 不是“一个更大的 prompt”。如果只是让模型从题目直接吐出证明，它会遇到两个瓶颈：自然语言证明难以自动评估；错误发生后缺少可恢复的局部反馈。形式系统

让 proof search 可以被切成许多小步，每一步都可检查。

AlphaProof 的关键接口

输入不是一个自由作文任务，而是一个形式数学任务。模型面对的是 Lean state，输出的是 Lean tactic 或形式化语句；环境返回新的 Lean state 或失败信息。这个接口决定了训练和推理都可以围绕“可验证状态转移”展开。

4.2 本章小结

AlphaProof 的根本设计不是“LLM 替代数学家”，而是“LLM/RL 在形式系统中探索”。这个设计把数学发现中的一部分过程转化为可搜索、可强化、可验证的计算过程。

5 IMO 2024：一次受约束的实战

讲座中段最重要的故事是 IMO 2024。Hubert 把这次参与称为 Apollo program：目标不是宣称系统已经掌握整个数学，而是在一个公开、困难、时间受限的环境里检验系统能做到什么。IMO 的价值在于题目难、评判标准明确、外部关注度高，而且题目在比赛时才公开。

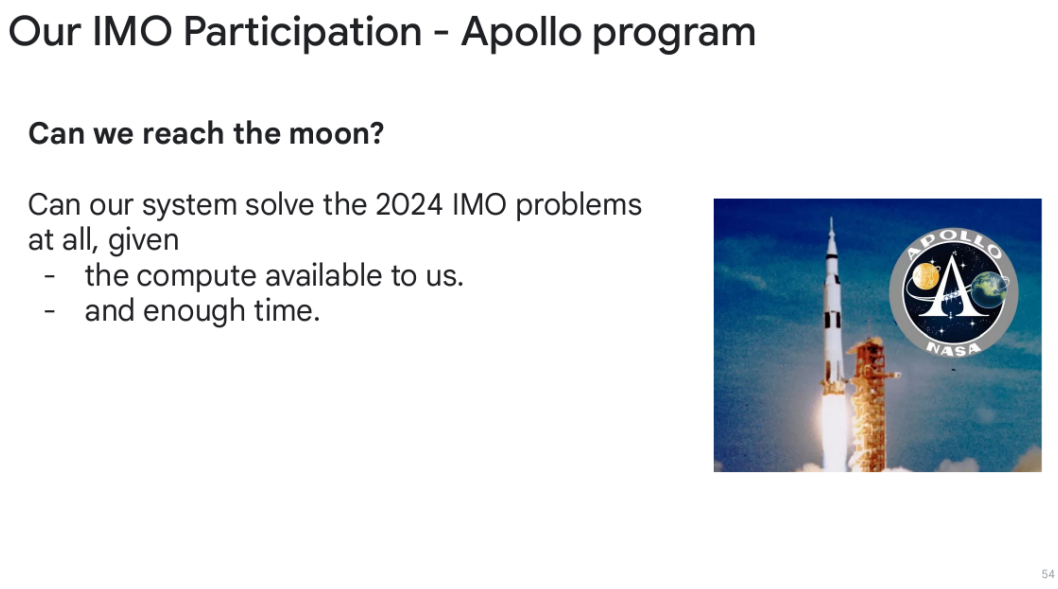


图 8: IMO 2024 被讲座称为 Apollo program：在真实竞赛约束中测试系统。图源：官方 slides p.54。

5.1 为什么几何要单独处理

2024 年 1 月，团队需要决定是否处理几何。限制非常具体：Mathlib 中二维欧几里得几何很稀疏，而 IMO 几何题需要这部分基础库。最终安排是 AlphaGeometry 处理几何，AlphaProof 处理代数、数论、组合等非几何题。这不是能力宣传上的小字，而是理解结果时必须保留的边界。

结果边界

讲座中明确把 P4 几何题归于 AlphaGeometry，而不是 AlphaProof。AlphaProof fully solved 的是 P1、P2、P6；P4 fully solved by AlphaGeometry。混写会夸大 AlphaProof 单系统能力。

5.2 正式协议

比赛题目正式发布后，团队协议如下：Lean 专家先手工形式化问题；Gemini 生成大量候选答案；Oracle 过滤数学上不可能或错误的候选；然后运行 test-time RL。这个协议说明 AlphaProof 并不是从自然语言原题直接端到端独立完成所有步骤。尤其是输入给系统的是已形式化问题，而非未经处理的自然语言题面。

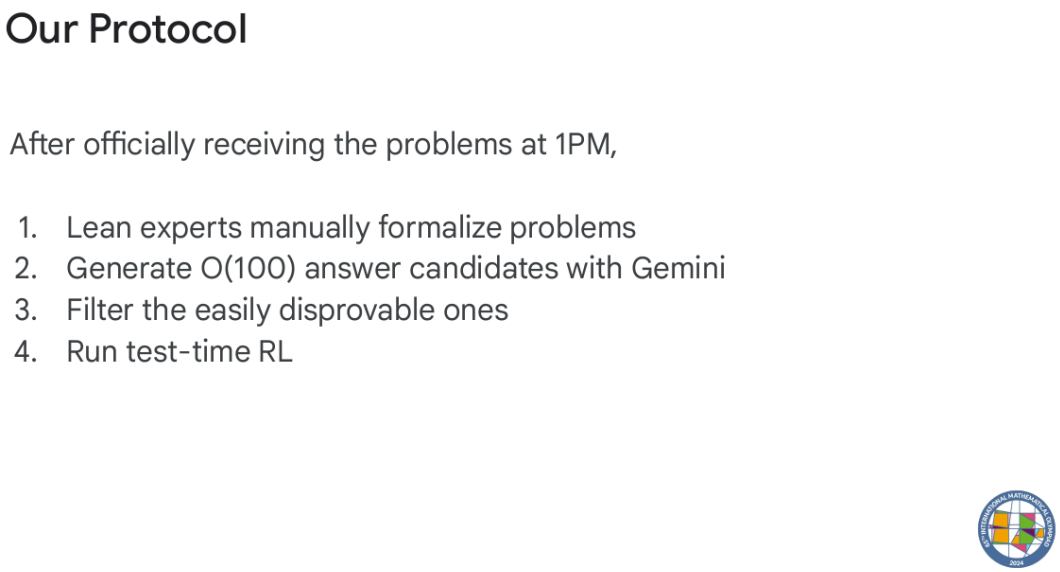


图 9: IMO 2024 协议：手工形式化、候选生成、Oracle 过滤、test-time RL。图源：官方 slides p.61。

5.3 结果与意义

最终，AlphaProof 完整解决 P1、P2、P6，AlphaGeometry 完整解决 P4。团队继续运行以期在 P3 上拿到额外分数，但进展不足以获得 partial point。Hubert 特别强调 P6 的难度：它被认为是近十年 IMO 中很难的问题之一，参赛者中只有极少数人完整解决。

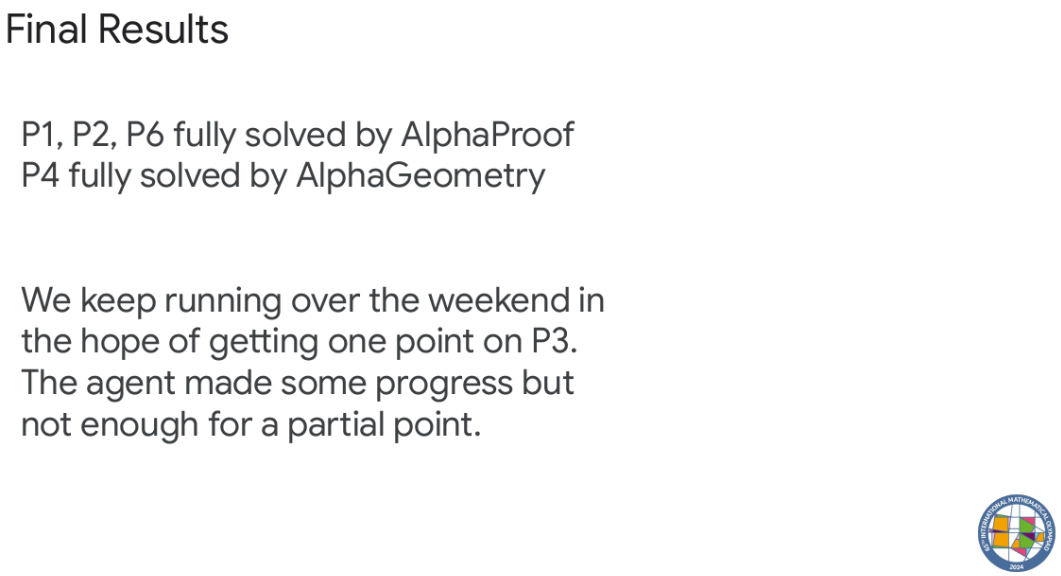


图 10: 最终结果：P1、P2、P6 由 AlphaProof 完整解决；P4 由 AlphaGeometry 完整解决。图源：官方 slides p.73。

IMO 故事真正证明了什么

它证明的是形式化输入、Lean 验证、搜索、RL 和 test-time adaptation 组合后，可以在真实高难数学竞赛中产生强结果。它没有证明端到端自然语言自动形式化已经解决，也没有证明 Mathlib 覆盖不足不再重要。

5.4 本章小结

IMO 2024 是 AlphaProof 的强证据，但它必须和协议一起读。手工形式化、几何分流、候选过滤和 test-time RL 都是结果的一部分。把这些边界保留下来，才能正确评估它对后续自动形式化和自动定理证明工作的启发。

6 AlphaProof 方法拆解

讲座后半段把 AlphaProof 方法拆成几个模块：formalizer model、prover model、AlphaZero-style search、autoformalization、基于 Mathlib 的监督训练、强化学习、以及 test-time RL。这一节是后续做自动形式化/ATP 最需要保留的部分。

6.1 Formalizer model：从自然语言到 Lean statement

Formalizer model 的任务是把自然语言描述的问题或定理转成 Lean 形式化。它解决的是“题目如何进入形式系统”的问题。AlphaProof 的 IMO 协议中，这一步仍由 Lean 专家手工完成；但在训练和问题扩展阶段，autoformalization 用来生成更多形式化问题。

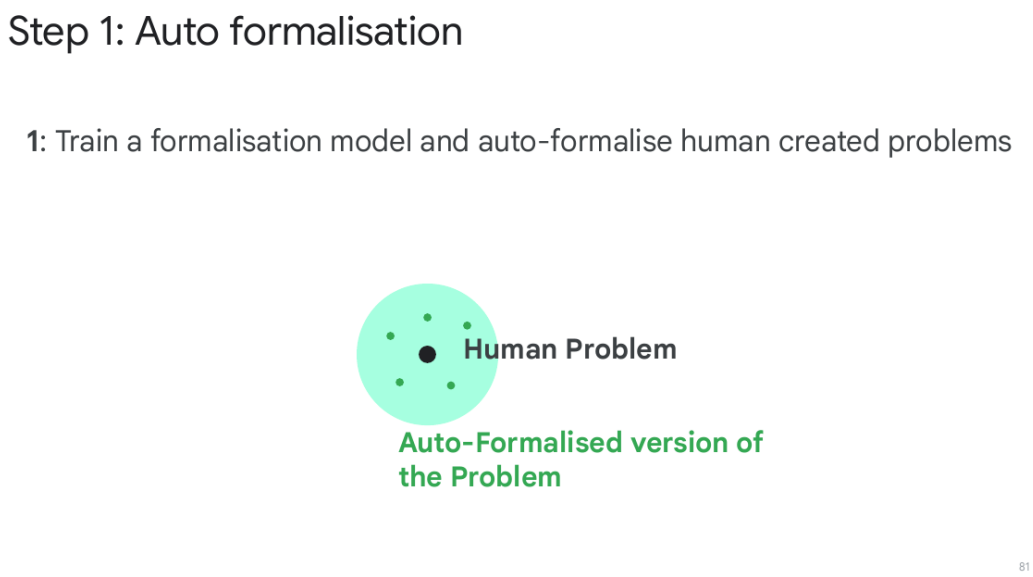


图 11: Step 1: 训练形式化模型，并自动形式化人类创建的问题。图源：官方 slides p.81。

自动形式化不是纯格式转换

自然语言数学经常省略条件、默认上下文和图形直觉。把题目转成 Lean statement 时，错误不一定表现为语法错误；更危险的是生成一个 Lean 可接受但语义不等价的题目。这也是下一讲 autoformalization 要重点讨论的问题。

6.2 Prover model：在 Lean state 上选 tactic

Prover model 面对的是 Lean state。它采样若干 tactic，把 tactic 送入 Lean，Lean 返回新的 state 或错误。如果证明完成，系统得到强验证信号；如果失败，搜索可以回退或换路。

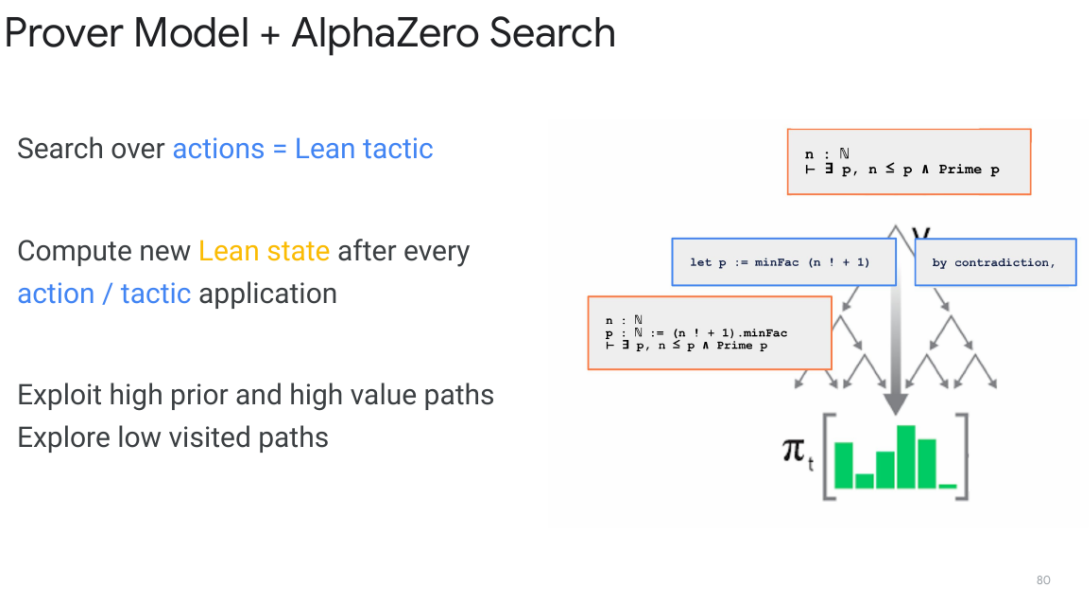


图 12: Prover model 加 AlphaZero search：把 proof search 看成状态和 tactic 的搜索树。图源：官方 slides p.80。

这个结构和自然语言 chain-of-thought 很不同。自然语言推理中，中间步骤即使看起来合理，也很难自动判定是否真的推进了证明；Lean tactic 则会真实改变 proof state 或失败。系统因此拥有可执行反馈。

6.3 Step 2：先从 Mathlib 学习证明动作

第二步是让 prover model 在 Mathlib 上做监督训练。slides 中给出的大致量级是约 100k definitions、200k theorems、300k lines of proofs。目标不是直接让模型解决 IMO，而是先让模型学会在 Lean 里采取合理动作，理解库中常见证明模式。

Mathlib 监督训练的角色

Mathlib 相当于“证明语言与库使用”的训练语料。它提供了大量正确证明轨迹，但这些轨迹并不等于 IMO 题训练集。Hubert 提到，如果只在 Mathlib 上训练，面对 IMO 风格问题的解决率仍接近 0，需要后续 RL。

6.4 Step 3：AlphaZero 强化学习

第三步是对 prover model 做 RL。对每个形式问题，系统生成用 Lean steps 搜索得到的经验，利用 Lean 验证证明是否正确，再通过强化学习更新 prover。这里的关键是“经验可以由系统自己产生”。随着训练推进，模型不只是模仿 Mathlib，而是在形式环境中探索新路径。

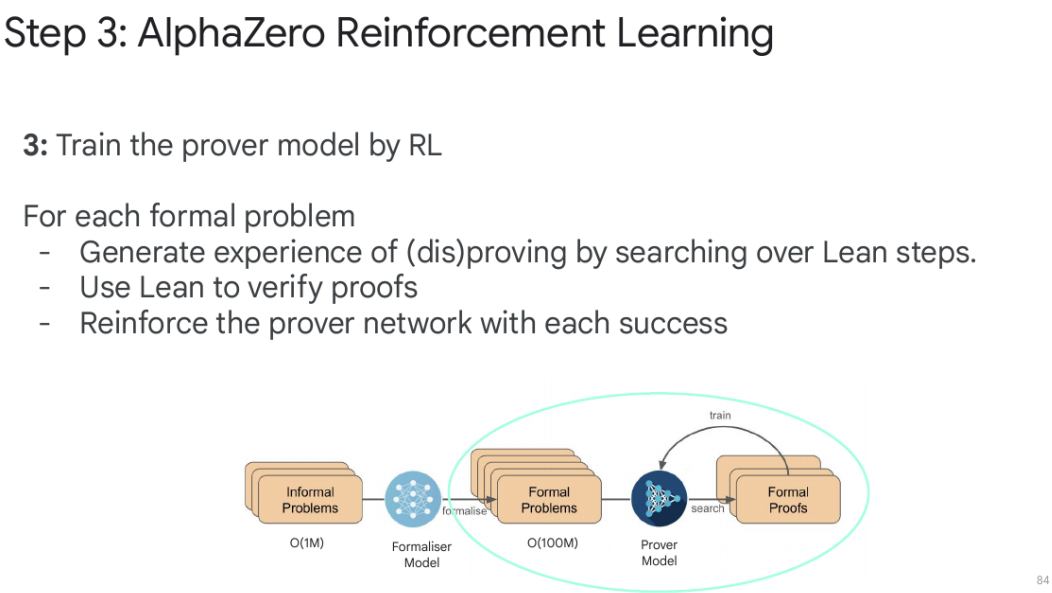


图 13: Step 3: 通过 AlphaZero 风格 RL 训练 prover model。图源：官方 slides p.84。

6.5 Final step：test-time RL

最后一步是 test-time RL：针对一个非常难的问题，围绕它生成变体或相关问题，训练一个 specialist checkpoint，再回到原问题上求解。这个思想很重要，因为 IMO 题目并不只是“从训练分布抽一题”。系统需要在推理时围绕目标题做局部适应。

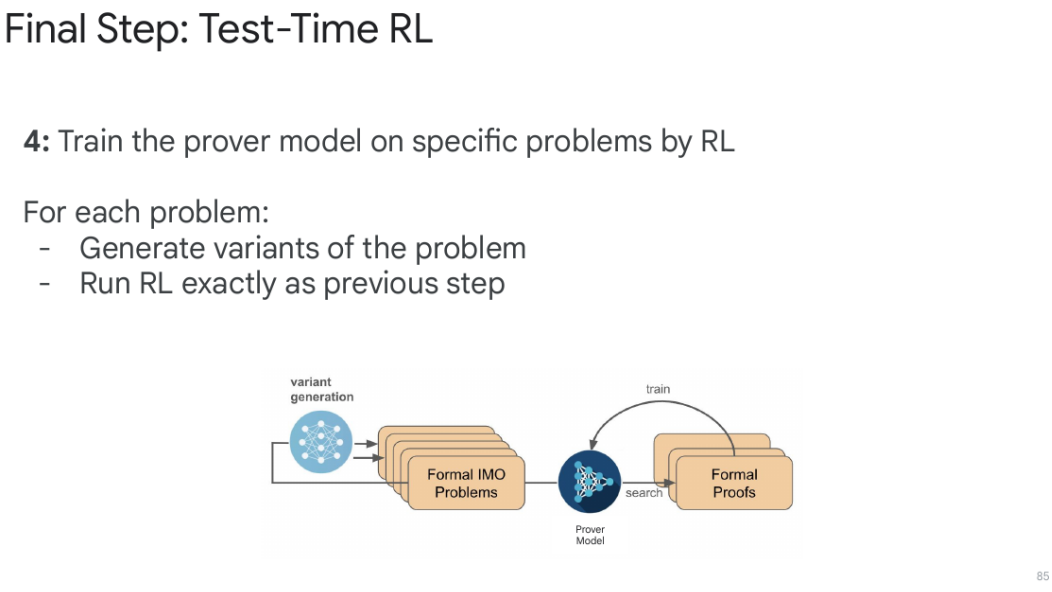


图 14: Final Step: 针对具体难题运行 test-time RL。图源：官方 slides p.85。

方法链条

AlphaProof 的方法链条可以概括为：自动/人工形式化问题 → Mathlib 监督训练 prover → Lean 环境中的 RL → 针对目标题的 test-time RL → 由 Lean 验证最终证明。

6.6 本章小结

AlphaProof 的技术核心不是单一模型，而是一条围绕 Lean feedback 建起来的训练和搜索管线。formalizer 解决“题目进入形式系统”，prover 解决“下一步怎么走”，RL 解决“如何从试错中提升”，test-time RL 解决“如何围绕具体难题适应”。

7 局限、失败模式与下一步

Hubert 对局限的表述很清楚。AlphaProof 继承了形式数学本身的挑战：大部分人类数学数据是自然语言；不是所有定理和领域都被 Mathlib 支持；数学中有大量“有趣性、美感、对象构造”的判断，不只是证明已有 statement。

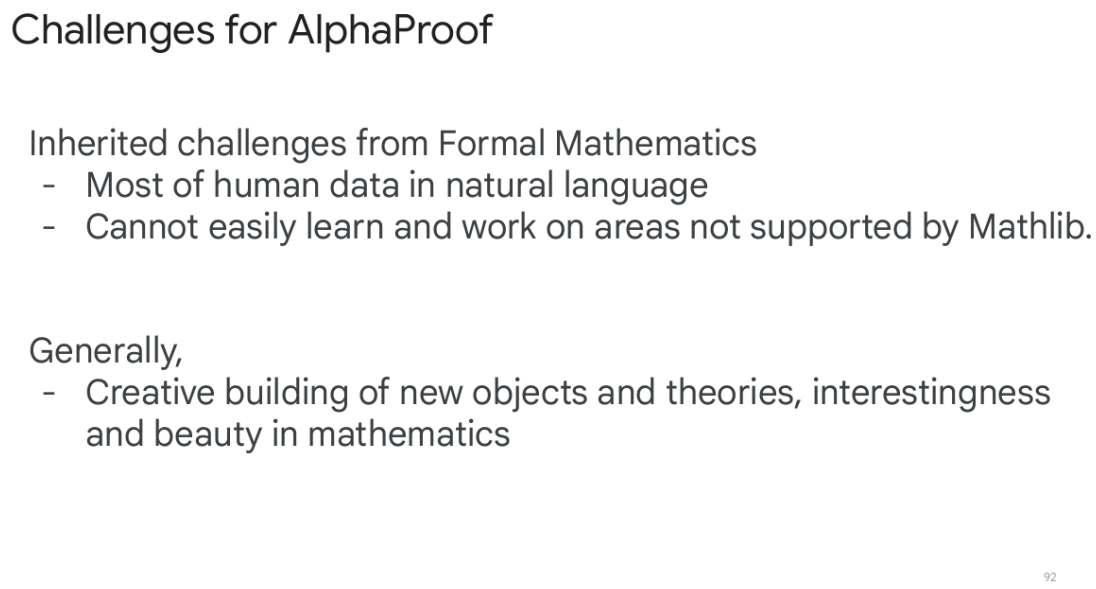


图 15: AlphaProof 继承了形式数学的挑战：自然语言数据、Mathlib 覆盖、创造性对象构造等。图源：官方 slides p.92。

7.1 Mathlib 空洞会直接变成能力边界

IMO 例子已经展示了这个问题：几何领域库不足导致 AlphaProof 不适合直接处理几何题，需要 AlphaGeometry 和专门几何基础设施。甚至在非几何题中，也会遇到 Mathlib 缺少某些定义或引理的问题。对后续工作来说，这意味着“prover 很强”不能替代“库足够可用”。

7.2 自然语言到形式语言仍是瓶颈

讲座把 autoformalization 放在 AlphaProof 管线第一步，但 IMO 实战中仍采用手工形式化。这说明自动形式化是核心方向，但不是已经完全解决的前置条件。对自动定理证明来说，形式 statement 的语义正确性比语法通过更重要。

7.3 下一步：从竞赛题走向数学全景

AlphaProof 的下一步被概括为：拓展到整个数学景观，贡献到研究数学前沿，并构建更强的数学 agent。这里的关键不是只把 IMO 分数继续提高，而是让系统能够在更广、更不标准化、更依赖定义

构造和库扩展的数学空间中工作。

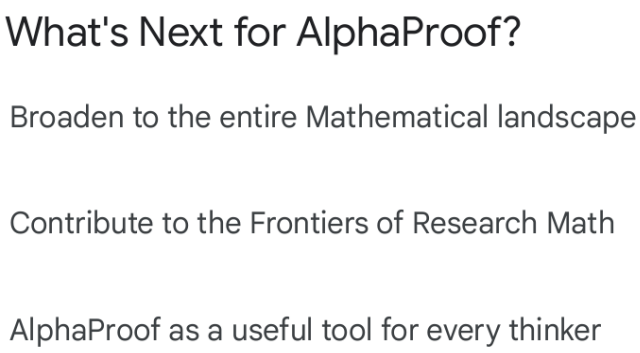


图 16: AlphaProof 的下一步：拓展到更广数学景观并走向研究数学。图源：官方 slides p.109。

最容易高估的地方

AlphaProof 展示了形式系统中 RL prover 的强潜力，但它没有消除三件事：形式化输入的语义风险、库覆盖不足、以及数学研究中“提出好问题/好定义”的开放性。

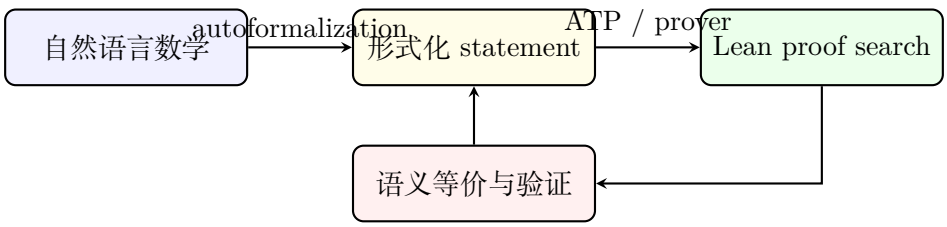
7.4 本章小结

AlphaProof 的局限不是边角料，而是下一步工作的地图。自动形式化、库扩展、检索、domain-specific reasoning、test-time adaptation 和 theorem proving search 都会在这些局限处相遇。

8 与下一讲的连接：自动形式化和 ATP

这节课为下一讲提供了一个实践案例：当形式化问题已经存在，Lean feedback 可以驱动证明搜索和强化学习。但下一讲 Kaiyu Yang 要讨论的正是更一般的问题：如何从自然语言数学走向形式定理和形式证明？如何训练 LLM 作为 theorem prover？如何评估自动形式化是否语义等价？以及当证明搜索空间无限大时，如何借助检索、符号工具和领域知识缩小动作空间？

可以把两讲的关系写成：



AlphaProof 更强调右半边：在形式化环境里证明。下一讲会补足左半边和中间连接：自动形式化的语义风险、证明搜索的动作空间、LeanDojo/ReProver 等系统，以及几何领域的特殊困难。

8.1 本章小结

如果你的下一步工作是自动形式化和自动定理证明，AlphaProof 最值得带走的是“环境化”的思想：不要只把 LLM 当答案生成器，而要围绕 proof assistant 设计输入、反馈、搜索、训练和验证闭环。

9 总结与延伸

Hubert 的收束不是简单说“系统拿了银牌水平”，而是强调一个更长期的方向：形式数学、软件工程工具和 RL/search 结合后，数学可能变成一个更可协作、更可验证、更可扩展的计算环境。AlphaProof 的 IMO 结果是一个重要节点，但不是终点。它证明了这个范式已经能在真实难题上工作，也暴露了通往通用数学 agent 的几个硬瓶颈。

从学习角度，本讲可以提炼为五个结论：

1. **形式系统是反馈机器**：Lean 的价值不只是最后验算，而是给每一步 proof search 提供状态和错误。
2. **RL 需要 grounded feedback**：数学证明之所以能接入 AlphaZero 式思想，是因为形式证明可以被检查。
3. **库覆盖决定可达空间**：Mathlib 的空洞会直接限制系统能探索什么。
4. **IMO 结果必须按协议解读**：手工形式化、AlphaGeometry 分流、Oracle 过滤、test-time RL 都是系统能力的一部分。
5. **下一步不是单纯扩大模型**：自动形式化、语义等价评估、检索增强 prover、领域专用工具和 test-time adaptation 都是核心。

给后续研究的操作性问题

做自动形式化/ATP 时，先问四个问题：输入 statement 是否忠实？proof assistant 给了哪些可利用反馈？搜索动作空间如何缩小？生成的证明或中间形式化能否被独立验证？AlphaProof 的贡献正是把这些问题组合成一个可运行的闭环。

9.1 建议继续追踪的问题

- 自动形式化如何评估“语义等价”，而不仅是 Lean 可编译？
- 当 Mathlib 没有足够引理时，系统应当扩展库、检索已有结果，还是改变 formal statement？
- test-time RL 的成本、稳定性和适用范围如何衡量？
- 研究数学中的“提出好定义”和“发现好问题”能否被形式环境给出足够反馈？

9.2 拓展阅读

- 课程主页：Berkeley CS294/194-280 Advanced Large Language Model Agents, Spring 2025
- 讲座 slides：AlphaProof: when RL meets Formal Maths
- 视频：Berkeley RDI YouTube recording