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

自动形式化与自动定理证明：Formal Reasoning Meets LLMs

cnfjlhj & Codex

2026 年 5 月 28 日

讲座：Language models for autoformalization and theorem proving
主讲：Kaiyu Yang, Meta FAIR
视频作者/频道：Berkeley RDI
发布日期：2025-04-13（课程日期 2025-04-07）
视频时长：52:07
视频链接：https://www.youtube.com/watch?v=cLhWEyMQ4mQ
课程页与 slides：课程主页；官方 slides

目录

1 这节课解决的不是同一个问题

上一讲 AlphaProof 展示了一个强系统：把问题放进 Lean，利用 proof assistant 的反馈做搜索和强化学习。这一讲 Kaiyu Yang 反过来问更基础的问题：为什么今天的 math/coding LLM 会成为能力竞赛中心？为什么只靠自然语言数学训练还不够？LLM 如何与 Lean 这样的形式系统结合成为 theorem prover？以及最难的，如何把非形式数学忠实地转换成形式数学？

因此这节课可以理解成 AlphaProof 的前置层和泛化层。AlphaProof 偏向“已有形式化问题以后，如何证明”；本讲覆盖“如何训练 theorem prover、如何评估 prover、如何自动形式化 theorem/proof、如何处理几何图形中的隐含推理”。

本讲主线

数学和代码之所以成为 LLM 训练焦点，是因为它们既代表复杂推理，又相对容易评估。但普通答案验证只能覆盖一部分能力；要走向高级数学和软件验证，需要把推理锚定到形式系统中。于是核心任务分成两类：自动定理证明，即在 Lean 中找 proof；自动形式化，即把 informal theorem/proof 变成等价的 formal theorem/proof。

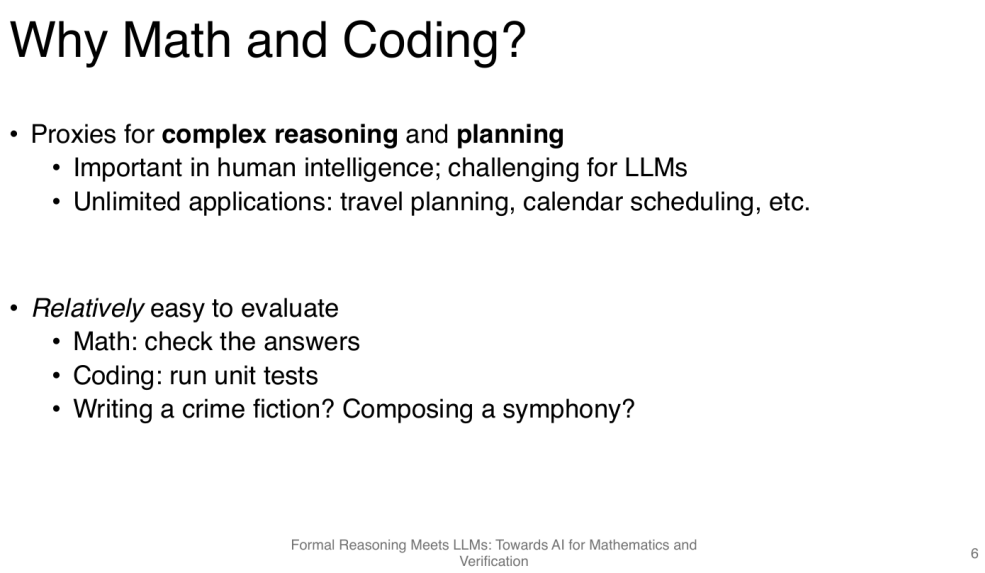


图 1: 数学和代码是复杂推理与规划的代理任务，同时相对可评估。图源：官方 slides p.6。

1.1 本章小结

这节课不是 AlphaProof 的重复，而是把“AI + formal math”的问题拆开：先看现代 math LLM 的训练路径，再看形式推理为什么是缺失环节，最后落到 theorem proving 与 autoformalization 两条技术线。

2 现代 Math LLM：SFT、RL 与可验证性

Yang 首先解释当前 LLM 数学能力竞赛的背景。OpenAI、Google、xAI 等模型发布时，经常突出 AIME、IMO、Codeforces 等数学和代码指标。原因不是这些任务覆盖了所有智能，而是它们既需要推理和规划，又比写小说、作曲等任务更容易定义评价标准。

2.1 SFT：好数据非常关键

监督微调的路径可以概括为：预训练模型先吸收文本和代码，再用数学网页、分步解题数据、工具集成解法等数据做后训练。这个路径的优势是直接、稳定、工程上成熟；限制是高度依赖数据质量和覆盖面。slides 提到公开数学数据集的规模可达约 900K，但高级数学、研究数学和高质量 proof 数据仍然稀缺。

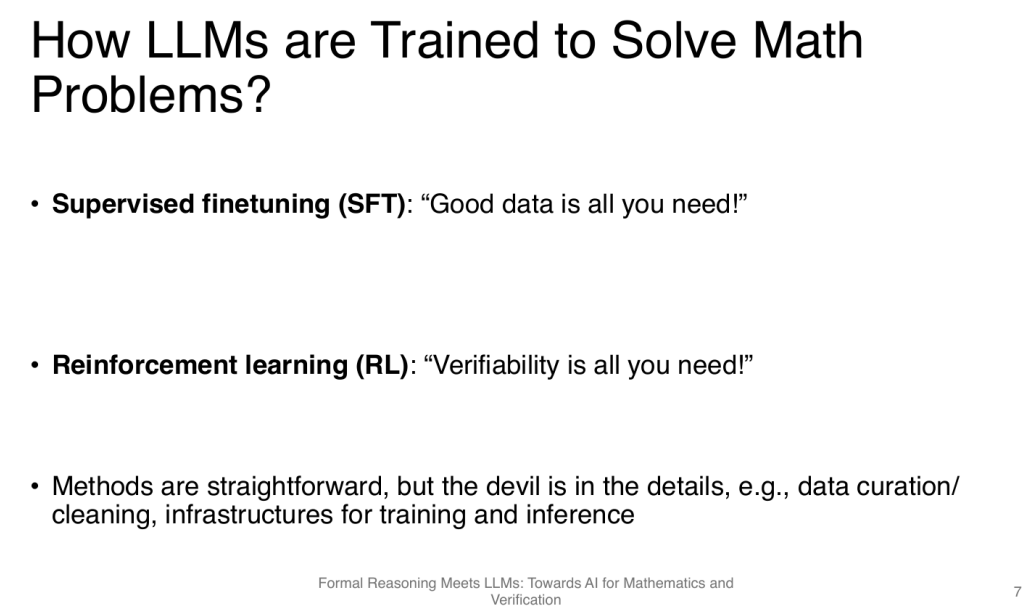


图 2: 数学 LLM 的两条常见训练口号：SFT 是“好数据很重要”，RL 是“可验证性很重要”。图源：官方 slides p.7。

2.2 RL：可验证题目让训练信号变硬

RL 路径的关键是 verifiable problems。如果题目有可检查答案，系统可以采样多个解法，只要最终答案正确就给奖励；对代码任务，可以跑 unit tests；对某些数学题，可以检查答案或用工具验证。这个思路解释了为什么 math/coding 任务在后训练中非常有吸引力。

但这也暴露出一个边界：答案正确不等于证明正确。对 pre-college math 或竞赛答案题，验证答案常常够用；对高级数学和证明写作，真正需要验证的是推理结构，而不是一个数值答案。

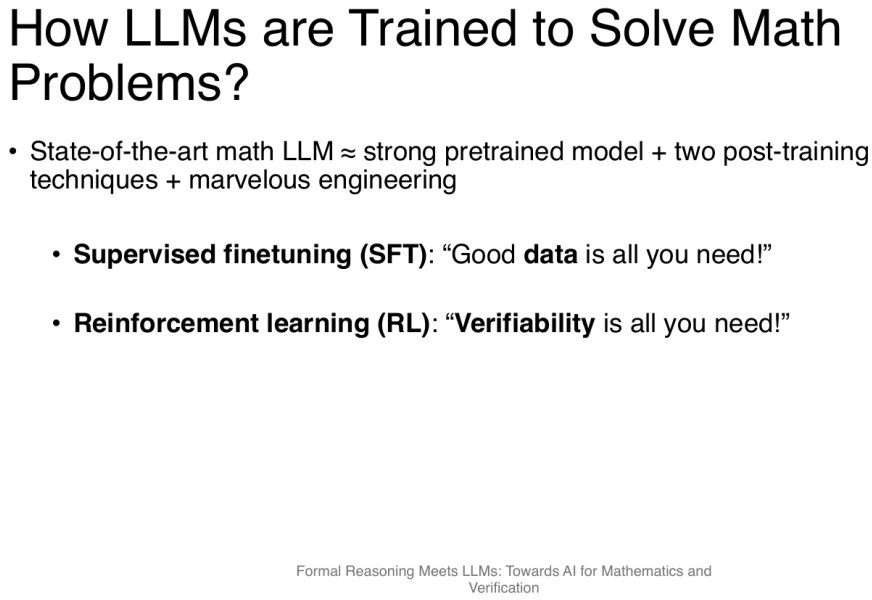


图 3: 强数学 LLM 通常来自强预训练模型加 SFT/RL 后训练；但这并不自动解决证明能力。图源：官方 slides p.23。

“可验证性”有层级

能验证最终答案，不等于能验证证明。能运行单元测试，也不等于能证明程序满足规格。形式推理的意义，是把评价从结果层推到推理结构层。

2.3 本章小结

SFT 和 RL 解释了当前数学 LLM 的快速进步：一个依赖高质量数据，一个依赖硬反馈。但它们在高级数学和证明任务上会遇到数据稀缺、推理不可直接验证、动作空间巨大等问题。这正是 formal reasoning 出场的原因。

3 为什么 LLM Alone 不够

Yang 用两类 gap 说明普通 math LLM 的不足。第一，从 pre-college math 到 advanced math。已有成功更多集中在 AIME、IMO 等竞赛任务，研究数学的数据更少、结构更开放。第二，从 guessing answers 到 writing proofs。模型可能给出正确答案或看似合理的证明，但证明是否有效仍需要形式检查。

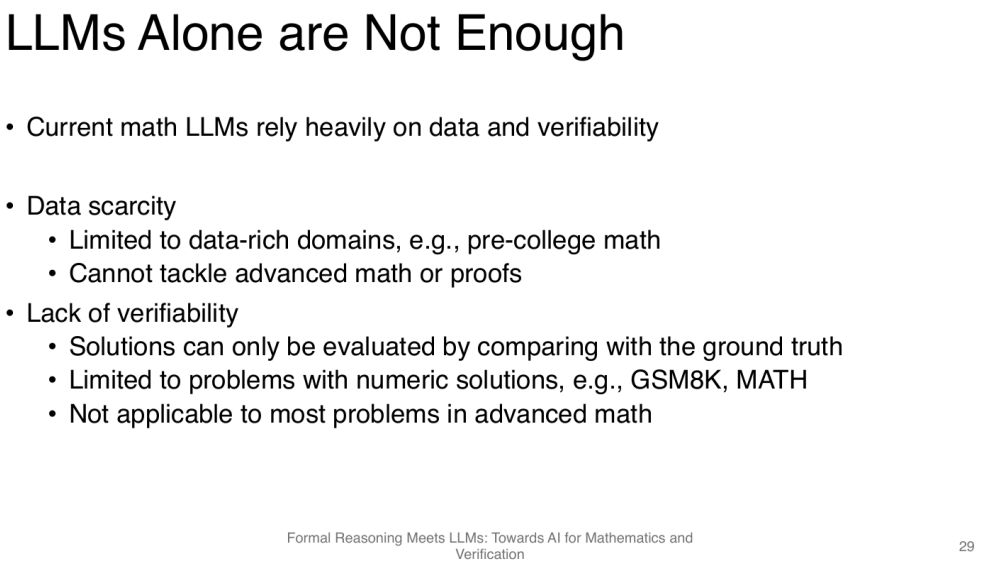


图 4: LLM alone 的限制：数据稀缺、验证不足、动作空间巨大。图源：官方 slides p.29。

3.1 缺失环节：Formal Reasoning

本讲给出的立场是：缺失环节是 formal reasoning，即把数学推理锚定在 Lean、Coq、Isabelle 等形式系统中。形式系统让 theorem/proof 变成机器可检查对象，并且可以在 proof search 过程中给出局部反馈。

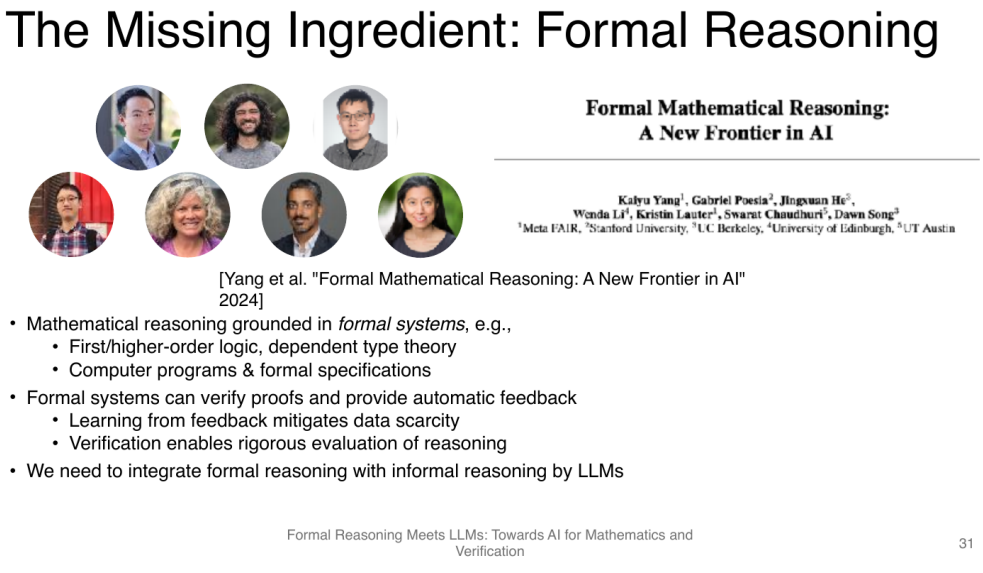


图 5: 形式推理被提出为 AI for mathematics 的新前沿。图源：官方 slides p.31。

从“答案正确”到“证明被核验”

自然语言数学模型可以通过采样、投票和工具辅助提高答案正确率；形式系统要求每一步都符合逻辑规则和库定义。二者不是替代关系，而是不同验证强度的层次。

3.2 Lean：把数学写成可检查程序

slides 用 Lean 文件展示 formalizing mathematics：一个 theorem statement、上下文、目标、proof tree、project 和依赖库共同构成形式证明环境。Lean 不是一个普通语法格式，而是一个交互式证明环境。模型要做 theorem proving，就需要在这个环境里选择下一步 tactic 或生成完整 proof。

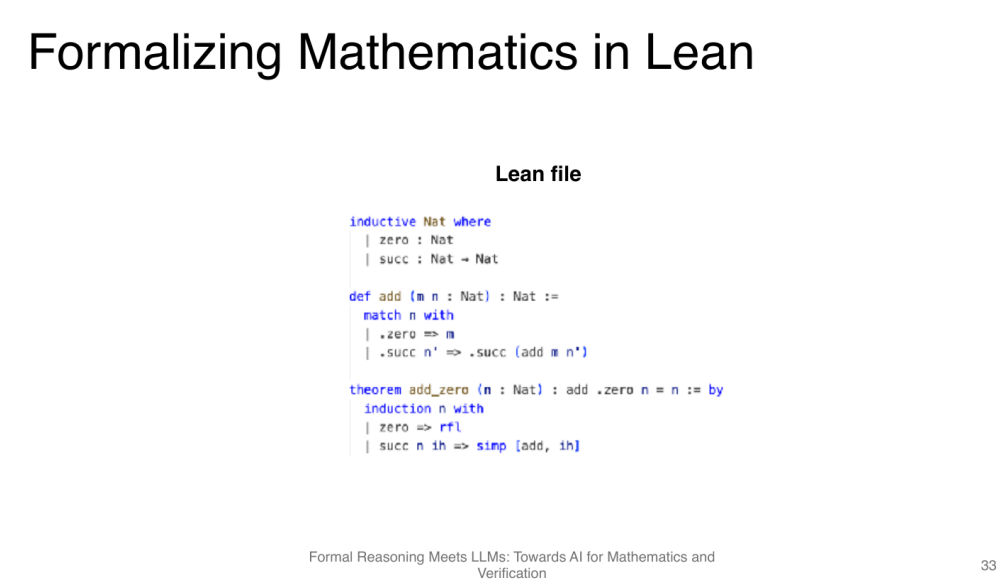


图 6: Lean 中的形式化数学：文件、局部上下文、目标和 proof tree 共同构成证明环境。图源：官方 slides p.33。

3.3 本章小结

LLM alone 的问题不是模型完全不会数学，而是缺少足够可靠的推理核验和结构化交互环境。形式系统提供了这个环境，使 theorem proving 可以被训练、搜索和评估。

4 LLM 做自动定理证明

自动定理证明在这节课中主要指：给定一个形式化 theorem 和当前 Lean proof state，模型生成下一步 tactic 或完整 proof，并与 Lean 交互直到证明完成。这里最重要的概念是“交互式证明”：模型不是一次性写完自然语言证明，而是在 proof assistant 中逐步推进。

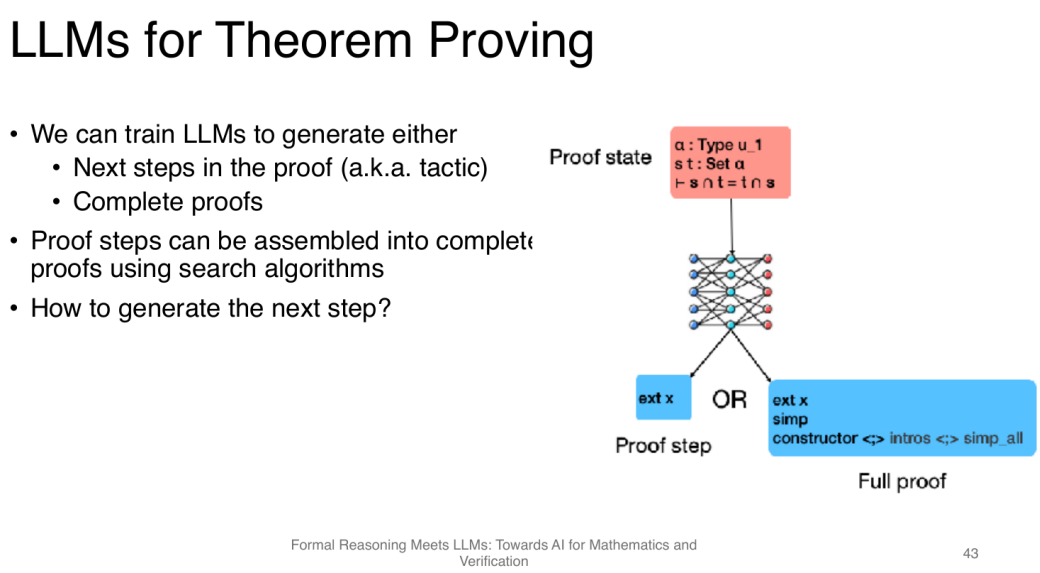


图 7: LLM theorem prover 可以生成下一步 tactic，也可以生成完整 proof。图源：官方 slides p.43。

4.1 LeanDojo：开放数据、工具和基准

Yang 介绍 LeanDojo，是因为早期很多 LLM prover 是私有系统，难以复现和比较。LeanDojo 提供开源的数据、训练/评估基准、模型 checkpoint，以及从 Lean 项目中抽取数据和与 Lean 交互的工具。slides 中给出的 LeanDojo Benchmark 包含 98,641 个 theorem/proof、217,639 个 tactic、129,162 个 premise。

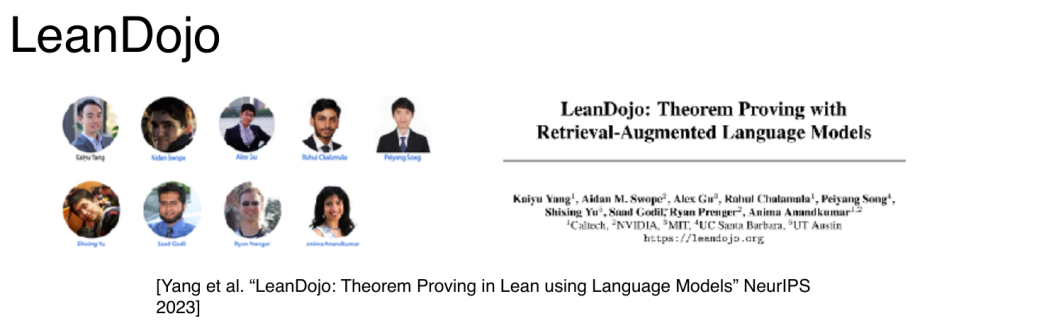


图 8: LeanDojo 为 Lean theorem proving 提供开放数据、模型和交互工具。图源：官方 slides p.46。

LeanDojo 的研究意义

它把“LLM 能不能在 Lean 里证明”从私有 demo 变成可训练、可评估、可复现的研究问题。对后续做 ATP，数据抽取、proof state 表示、premise 访问和 Lean 交互接口都比单纯模型大小更基础。

4.2 ReProver：检索增强 theorem prover

ReProver 的核心思想是：给定当前 proof state，从所有可访问 premises 中检索相关前提，把检索结果和 state 拼接后用于 tactic generation。这里的检索不是普通 RAG 的装饰，而是在 formal proof 中缩小可用知识空间。一个证明状态可调用的 lemma 太多，如果不检索，模型很难在巨大库中找到合适工具。

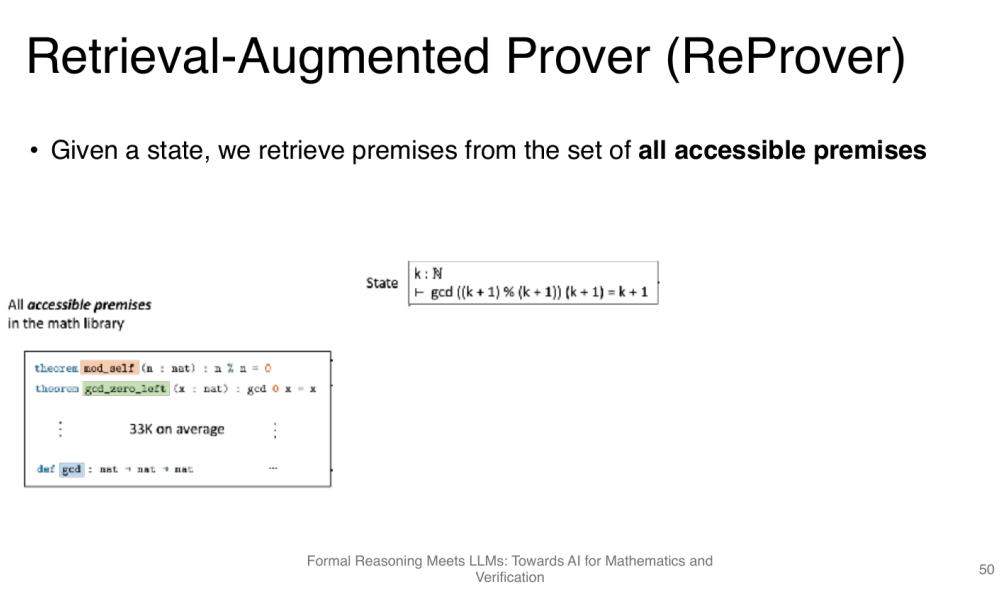


图 9: ReProver：根据当前 state 检索可访问 premise，再生成 tactic。图源：官方 slides p.50。

4.3 典型 neural theorem prover 的结构

一个典型 neural theorem prover 可以概括为：从 theorem 出发，维护 proof search tree；每个节点是 Lean state；模型根据 state 和检索到的 premises 生成 tactic；Lean 执行 tactic 后返回新 state；搜索策略决定继续扩展哪个节点。

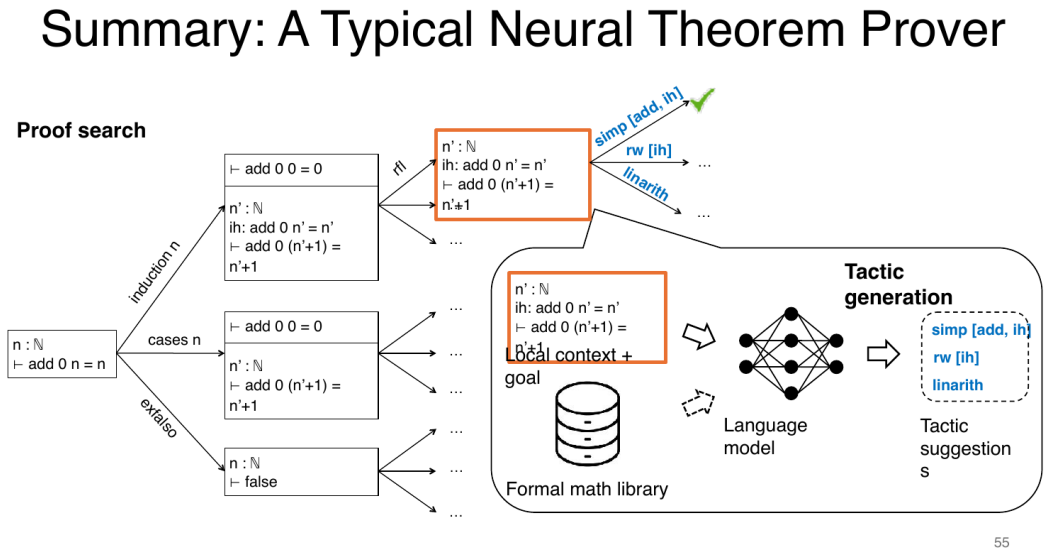


图 10: 典型 neural theorem prover：proof search、tactic generation、Lean interaction 和 formal library 结合。图源：官方 slides p.55。

不要把 theorem proving 简化成文本生成

模型输出的是 tactic 或 proof code，但真正的任务是状态空间搜索。Lean 返回的新目标、失败信息和可访问 premise 决定下一步策略。如果忽略交互和搜索，只看生成文本，会低估 ATP 的系统性难度。

4.4 本章小结

LLM theorem proving 的关键不是让模型“像人一样写证明”，而是让模型成为 proof search 中的策略组件。LeanDojo 提供开放环境，ReProver 展示检索如何缩小知识空间，Lean 交互则提供可靠反馈。

5 动作空间问题：从无限搜索到领域约束

Yang 接着指出 theorem proving 的一个核心限制：动作空间巨大，甚至可以说近似无限。围棋虽然复杂，但棋盘和合法动作有限；数学证明中的 tactic、lemma、变形、定义展开、构造选择和中间命题几乎无穷。人类数据无法均匀覆盖这个空间，RL 探索也会变得困难。

5.1 LIPS：用领域结构约束不等式证明

为了解释如何缩小动作空间，讲座介绍 LIPS，即 LLM-based Inequality Prover with Symbolic Reasoning。它面向 Olympiad-level inequalities，把不等式证明中的步骤分成更有结构的类型，例如 rewriting 和 scaling，并让 LLM 与 symbolic tools 分工：LLM 负责提出重写方向，符号工具负责缩放或验证相关变换。

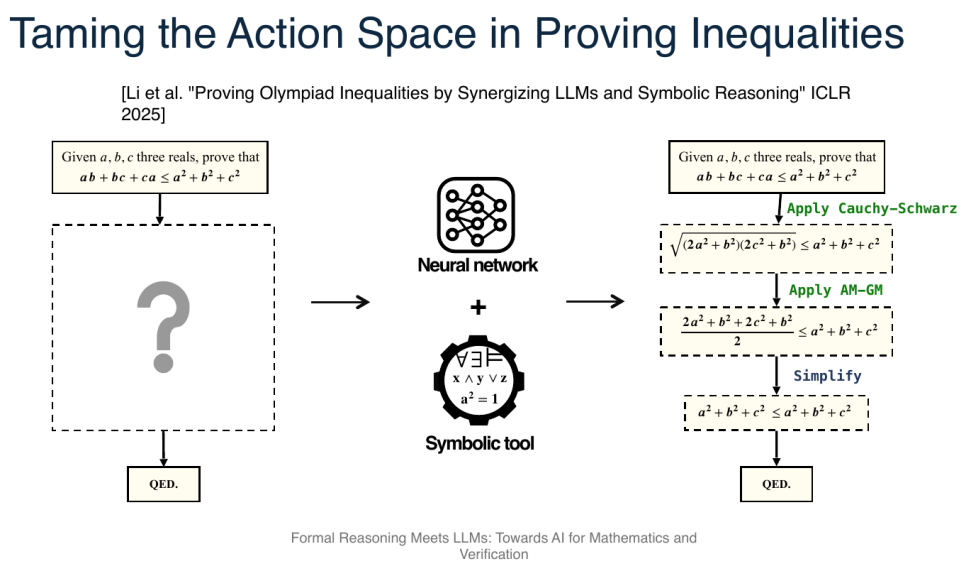


图 11: 不等式证明展示了巨大动作空间：可能的变形、引理和中间不等式非常多。图源：官方 slides p.58。

Experimental Results

- Our system LIPS surpasses IMO Gold Medalists in inequality proving

	DeepSeek-R1	Gold medalists	LIPS
#Solved Olympiad-level Inequalities*	4/20	15/20	**16/20**

* Problems are collected from IMO competitions, national team selection test, training quizzes.

- LIPS achieves SoTA performance across various competition-level datasets

Dataset	# of Problems	Neural Provers			Symbolic Provers		LIPS	Δ
		DSP	MCTS	AIPS†	CAD†	MMA†		
ChenNEQ	41	0.0	17.0	-	70.7	68.2	**95.1**	24.4↑
MO-INT	20	0.0	15.0	50.0	60.0	60.0	**80.0**	20.0↑
567NEQ	100	0.0	4.0	-	54.0	52.0	**68.0**	14.0↑
Total	161	0.0	8.6	-	59.0	57.1	**76.3**	17.3↑

† The code of AIPS has not been publicly available, we only include its originally reported results.
‡ CAD and MMA only output verification results, they cannot produce human-readable proofs.

图 12: LIPS 在 Olympiad-level inequalities 上的结果：20 题中解决 16 题。图源：官方 slides p.64。

LIPS 的一般启发

当通用 prover 面对无限动作空间时，领域知识不是落后手工规则，而是搜索结构。把动作空间分解成可控操作类别，常常比盲目扩大采样更有效。

5.2 本章小结

自动定理证明的难点不只是“模型聪不聪明”，还包括如何组织搜索空间。LIPS 的启发是：在特定数学领域里引入符号工具和领域动作，可以把不可搜索的问题变得可搜索。

6 Autoformalization：从 informal 到 formal

自动形式化是本讲后半段的重心。slides 区分了两类任务：

- **Theorem autoformalization**：informal theorem → formal theorem。
- **Proof autoformalization**：informal theorem/proof + formal theorem → formal proof。

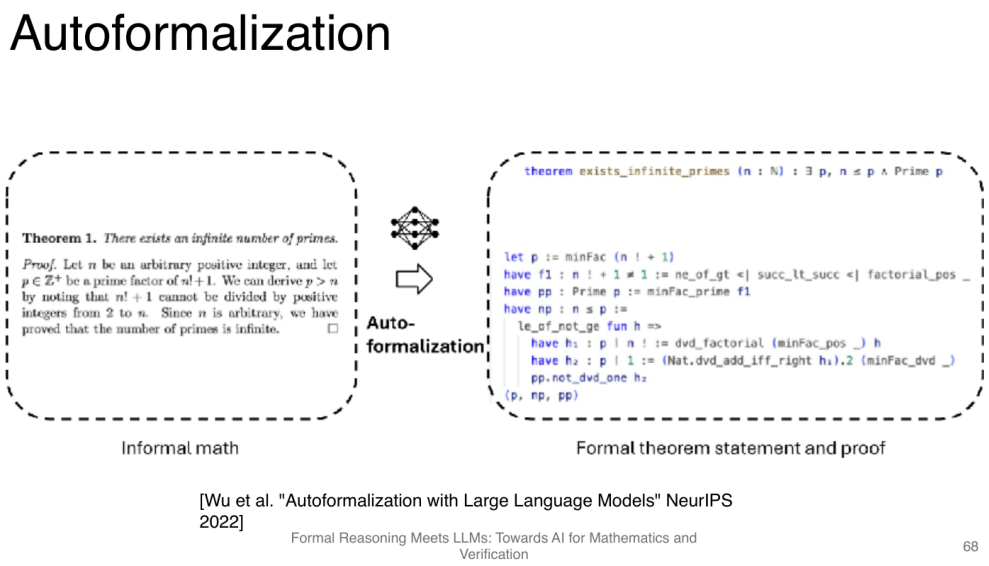


图 13: Autoformalization 的基本问题：把非形式数学转换成形式数学。图源：官方 slides p.68。

6.1 为什么 theorem autoformalization 难评估

定理自动形式化最棘手的问题是：没有可靠自动评价。一个生成的 Lean statement 可以通过语法检查，也可能类型正确，但它是否和原始自然语言 theorem 等价，并不容易自动判断。理论上可以做 equivalence checking：证明两个 theorem 互相推出。但一般情况下这不可行，因为你又回到了 theorem proving 本身。

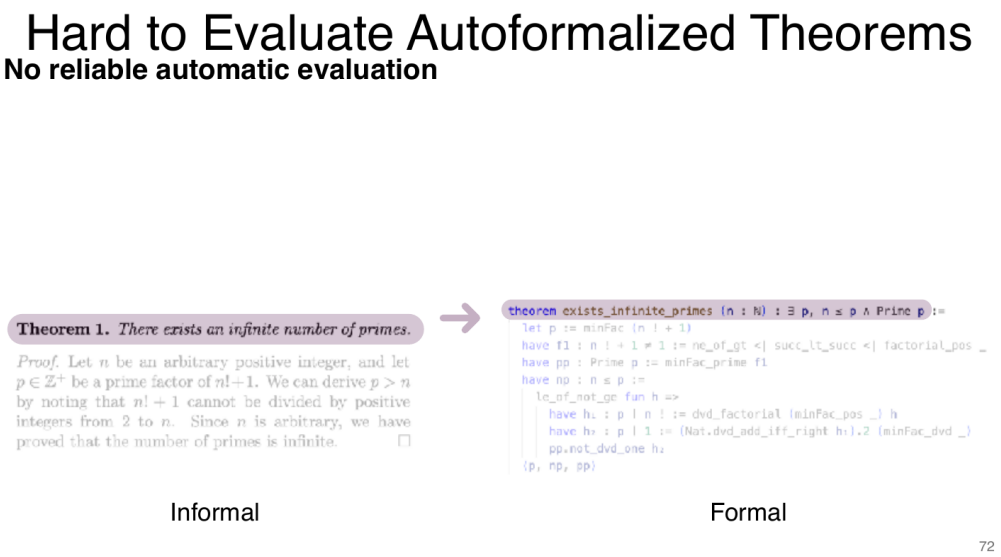


图 14: 自动形式化 theorem 难以自动评估：语法通过不代表语义等价。图源：官方 slides p.72。

人工评价又昂贵，尤其当 statement 涉及复杂定义、隐含条件或领域约定时。于是 autoformalization 研究中常见的通过率、类型检查率、BLEU/相似度等指标都只能反映一部分，不足以证明忠实性。

最危险的成功样子

一个 autoformalized theorem 如果 Lean 接受，但丢了条件、改了量词、弱化了结论，它会表现得像成功样本，却在语义上失败。后续研究必须把这种“可编译但不忠实”的错误单独审计。

6.2 为什么 proof autoformalization 难

proof autoformalization 的主要困难是 informal proofs 有 reasoning gaps。自然语言证明常写“显然”“类似可得”“留给读者”，也依赖读者从图形、上下文或常识补全细节。形式证明必须 gap-free，所有隐含步骤都要展开到 proof assistant 能检查的程度。

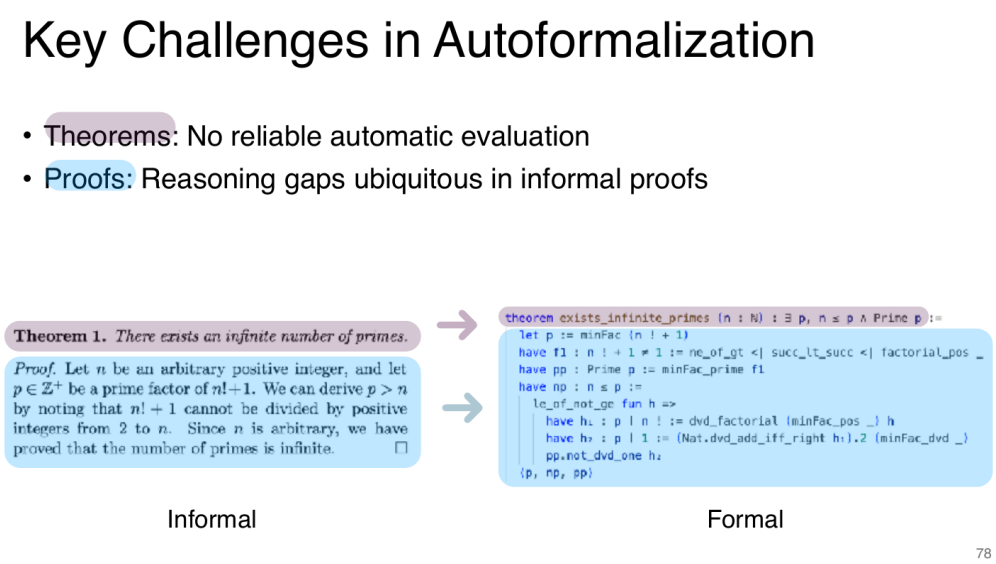


图 15: 自动形式化的两个核心挑战：theorem 难评价，proof 充满 reasoning gaps。图源：官方 slides p.78。

自动形式化的两类失败

Theorem 失败常表现为“生成了一个不等价的形式 statement”；Proof 失败常表现为“原证明缺口太多，无法直接翻译成 gap-free formal proof”。这两类失败需要不同的检测和修复策略。

6.3 本章小结

Autoformalization 不是把英文变成 Lean 的机器翻译。它涉及语义等价、隐含条件、证明缺口、库定义选择和领域表示。它是 ATP 的前置环节，也是最容易隐藏错误的环节。

7 欧几里得几何：图形推理与隐含缺口

Yang 用 Euclidean geometry 展示 autoformalization 的困难，因为几何是人类和机器智能的经典 arena，也因为几何证明高度依赖 diagrammatic reasoning。讲座介绍 LeanEuclid：一个用于自动形式化欧几里得几何的 benchmark，包含来自 Euclid's Elements 和 UniGeo 的问题。

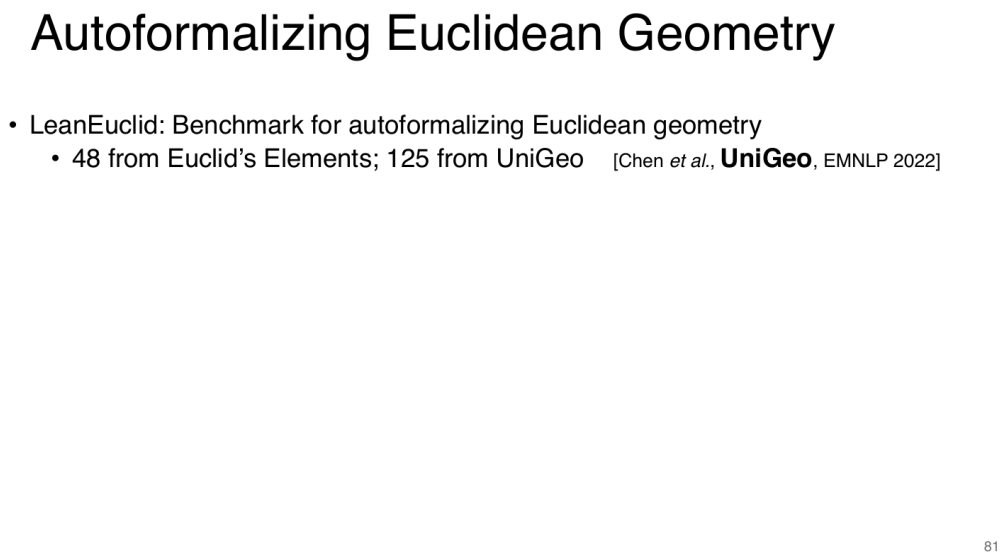


图 16: LeanEuclid：面向欧几里得几何自动形式化的 benchmark。图源：官方 slides p.81。

7.1 逻辑缺口：图上看见的不等于 Lean 知道

几何证明常依赖图。人看到两个圆相交、点在某条线上、三角形方向关系，就自然补全了许多事实。但 Lean 不看图，也不会自动知道“这个点在这个区域”“这两个对象按图形位置相交”。形式化时，必须把这些关系显式写出来，或用可证明的构造替代图形直觉。

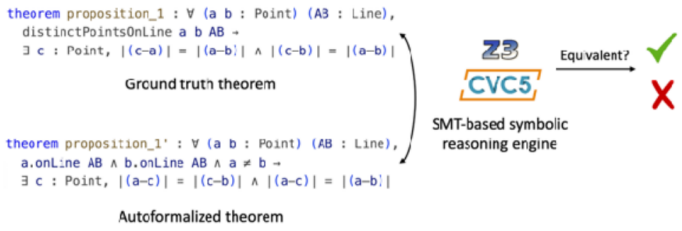
讲座用 Euclid's Elements Book I Proposition 24 说明：非形式证明里存在多个隐含步骤，形式化时必须补出等腰三角形、角相等、角大小关系等推理。图形帮助人理解，但机器需要逻辑对象。

7.2 等价检查与符号引擎

对于几何 theorem，讲座还介绍了 theorem equivalence checking：两个 theorem 等价，当且仅当可以证明 $T_1 \Leftrightarrow T_2$。在特定几何设置中，可以借助 SMT solver 等符号推理引擎做一部分检查。这是把一般不可行的问题放到特定领域里变得可处理的例子。

Equivalence Checking Between Theorems

- Two theorems T_1 and T_2 are equivalent iff we can prove $T_1 \iff T_2$
- Symbolic reasoning engine based on SMT solvers



98

图 17: Theorem equivalence checking：在特定形式系统和符号推理引擎中检查两个 theorem 是否等价。图源：官方 slides p.98。

7.3 Diagrammatic reasoning 如何形式化

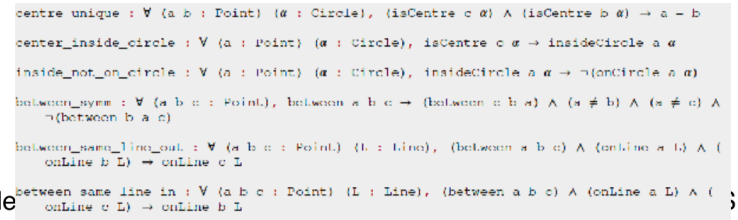
slides 提到 Formal System E，用于刻画 Euclid's Elements 中的图形推理。核心思想是：图形推理可以被当成逻辑规则建模，并与 SMT solver 结合。这样，几何图上的某些“看得见”的关系可以转化成形式系统内的可推理对象。

Modeling Diagrammatic Reasoning

The Formal System E

[Avigad *et al.*, "A formal system for Euclid's Elements", 2008]

- Diagrammatic reasoning are logical consequences of "diagrammatic rules"



- We imple SMT solvers

105

图 18: Formal System E：把欧几里得几何中的 diagrammatic reasoning 建模为逻辑规则。图源：官方 slides p.105。

几何为什么是好测试场

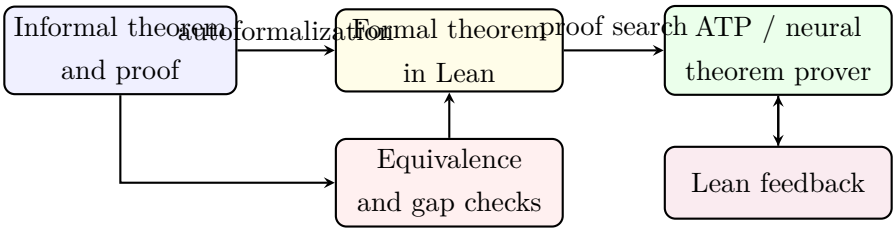
几何同时包含自然语言、图形、符号、隐含构造和证明缺口。它迫使 autoformalization 系统同时处理 statement 等价、proof gap filling 和 diagram-to-logic 的转换，因此比纯代数题更暴露系统弱点。

7.4 本章小结

欧几里得几何显示了 autoformalization 的本质困难：人类证明依赖大量未显式写出的图形事实，而形式证明要求这些事实全部进入逻辑系统。解决几何，需要把图形推理、符号推理和 Lean formalization 放在同一条管线里。

8 把两条线合起来

到这里，本讲的两条线可以合并：



Theorem proving 需要形式化输入；autoformalization 需要可靠评价；评价又常常需要 theorem proving 或领域符号系统。这就是为什么这个方向不能被拆成一个简单的“翻译模型”和一个“证明模型”。它更像一条闭环系统：生成、检索、证明、验证、修复互相依赖。

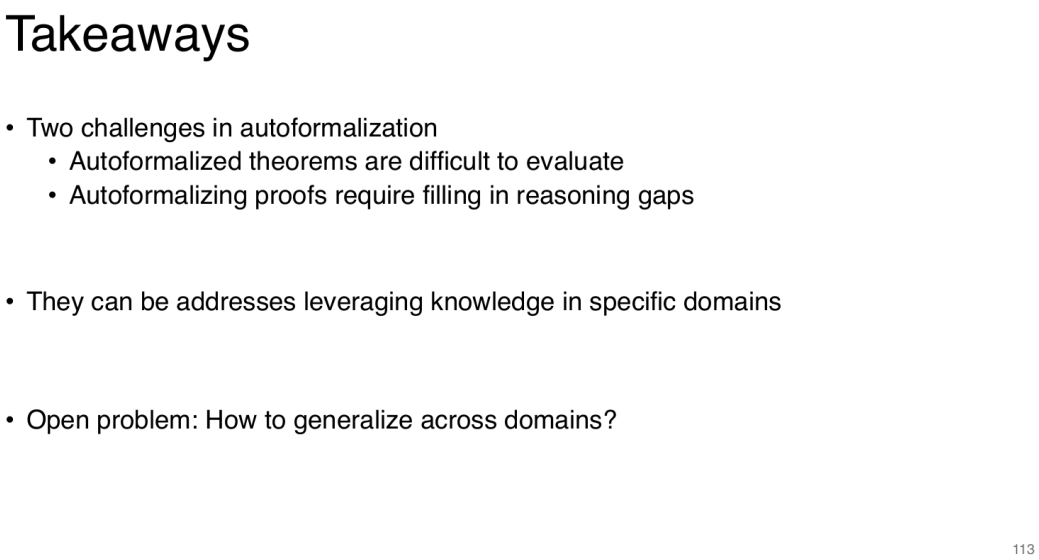


图 19: 讲座 takeaways：autoformalization 的 theorem 评价和 proof gap 是核心挑战，可借助特定领域知识处理。图源：官方 slides p.113。

8.1 本章小结

本讲最后给出的 takeaways 很克制：autoformalized theorems 很难评价，autoformalizing proofs 需要填补 ubiquitous reasoning gaps；这两者可以借助特定领域知识处理，但开放问题是如何跨领域泛化。

9 总结与延伸

这节课的实用价值在于，它把 “LLM 做数学” 拆成了可操作的研究接口。SFT/RL 提供当前 math LLM 的基础；formal reasoning 提供更强验证层；LeanDojo/ReProver 展示 neural theorem prover 的数据、检索和交互结构；LIPS 展示领域约束如何缩小动作空间；autoformalization 和 LeanEuclid 展示自然语言/图形数学进入形式系统时的语义风险。

从后续工作角度，最重要的结论有五个：

1. **答案可验证不是证明可验证**：普通 math RL 的奖励信号不足以覆盖高级证明。
2. **ATP 是系统工程**：模型、检索、proof search、Lean 交互和库依赖共同决定结果。
3. **动作空间必须被结构化**：领域工具和符号推理能把无限搜索空间变窄。
4. **Autoformalization 的核心是忠实性**：可编译 statement 不是成功，语义等价才是关键。
5. **几何暴露了隐含推理**：diagrammatic reasoning 是从人类证明到形式证明时必须处理的难点。

给自动形式化/ATP 项目的落地检查表

每个实验至少要明确四件事：formal statement 的来源和忠实性检查；proof search 使用的 premise 检索或动作约束；Lean 失败时的错误分类；最终证明是否由 proof assistant 独立检查。没有这些，结果很容易退化成 “模型输出看起来像证明”。

9.1 建议继续追踪的问题

- 如何构造比 type-check 更强、比人工评审更便宜的 autoformalization 评价?
- ReProver 式 premise retrieval 在不同数学领域的瓶颈是什么?
- LIPS 的领域动作分解能否迁移到组合、数论、拓扑等领域?
- 几何中的 diagrammatic reasoning 能否作为多模态 formal reasoning 的通用测试场?
- AlphaProof 的 test-time RL 如何和本讲的 LeanDojo/ReProver/autoformalization 管线合并?

9.2 拓展阅读

- 课程主页：Berkeley CS294/194-280 Advanced Large Language Model Agents, Spring 2025
- 讲座 slides：Formal Reasoning Meets LLMs: Towards AI for Mathematics and Verification
- 视频：Berkeley RDI YouTube recording