DeepSeek-Prover-V1.5: 利用证明助手反馈进行强化学习与蒙特卡洛树搜索

Huajian Xin*, Z.Z. Ren*, Junxiao Song*, Zhihong Shao*, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang
Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z.F. Wu, Fuli Luo, Chong Ruan

DeepSeek-AI

https://github.com/deepseek-ai/DeepSeek-Prover-V1.5

Abstract

我们介绍了 DeepSeek-Prover-V1.5，这是一个专为 Lean 4 定理证明设计的开源语言模型，它通过优化训练和推理过程对 DeepSeek-Prover-V1 进行了增强。该模型在 DeepSeekMath-Base 上进行预训练，并专注于形式化数学语言，随后使用基于 DeepSeek-Prover-V1 构建的增强型形式化定理证明数据集进行监督微调。进一步的性能提升是通过基于证明助手反馈的强化学习（RLPAF）实现的。除了 DeepSeek-Prover-V1 的单次完整证明生成方法外，我们提出了 RMaxTS，这是一种蒙特卡洛树搜索的变体，采用内在奖励驱动的探索策略来生成多样化的证明路径。DeepSeek-Prover-V1.5 相较于 DeepSeek-Prover-V1 取得了显著改进，在高中水平的 miniF2F 基准测试集上达到了新的最先进水平（63.5%），在本科水平的 ProofNet 基准测试集上也取得了新的最先进结果（25.3%）。

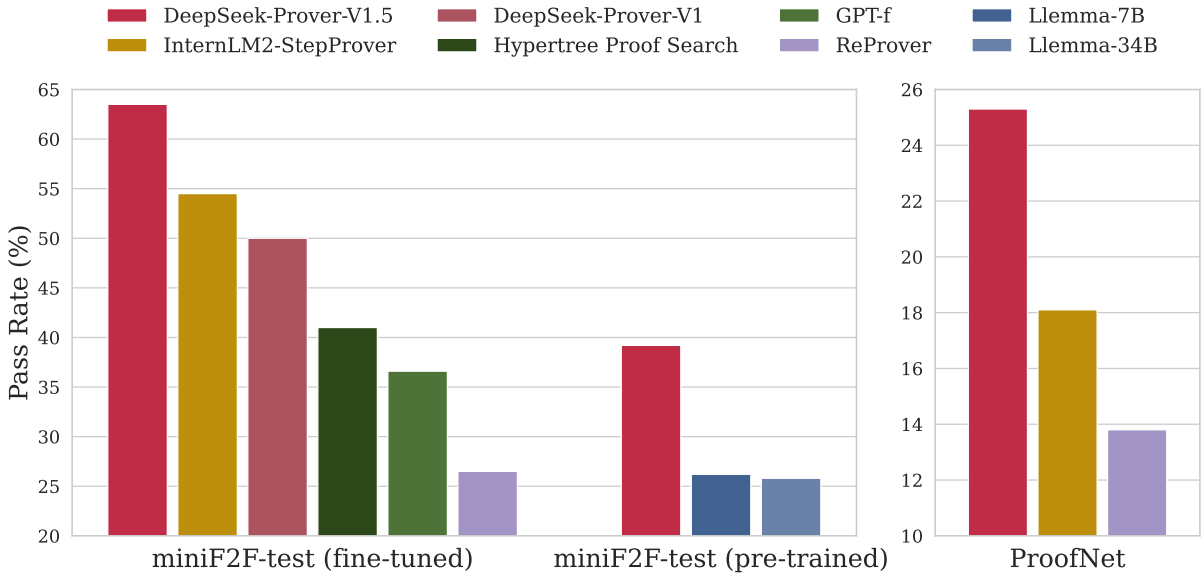


图 1 | 模型在 Lean 4 形式化定理证明基准测试上的通过率：高中水平的 miniF2F-test 基准测试 (Zheng et al., 2022) 和本科水平的 ProofNet 基准测试 (Azerbayev et al., 2023)。我们将 DeepSeek-Prover-V1.5 的预训练版本和微调版本与强大的基线模型进行了比较。

* 核心贡献者

1. 引言

大型语言模型的近期进展对人工智能领域的数学推理与定理证明产生了深远影响。尽管在自然语言领域取得了显著进展，但语言模型在形式化定理证明（*e.g.* 使用 Lean (Moura and Ullrich, 2021) 和 Isabelle (Paulson, 1994)）中仍面临巨大挑战，这需要满足验证系统形式化规范的严格推导。即使是 GPT-4 (OpenAI, 2023) 等先进模型在处理复杂形式化证明时也显得力不从心，这凸显了其中涉及的编程与数学的复杂性。形式化定理证明模型不仅需要掌握 Lean 定理证明器等形式系统的语法与语义，还需将抽象的数学推理与精确的形式化表示相统一。

形式化定理证明中的语言模型通常采用两种策略：证明步骤生成 (Polu and Sutskever, 2020; Jiang et al., 2022; Lample et al., 2022; Yang et al., 2023; Wu et al., 2024) 与完整证明生成 (Jiang et al., 2022; Zhao et al., 2023; Wang et al., 2023)。证明步骤生成预测每一个后续战术（tactic），并使用形式化验证器进行验证以获取当前战术状态的更新信息，通常利用树搜索技术来构建有效证明。相比之下，完整证明生成计算效率更高，它基于定理陈述直接生成完整的证明代码，减少了证明模型与形式化定理验证器之间协调所需的通信开销。尽管 DeepSeek-Prover-V1 (Xin et al., 2024) 通过完整证明生成在 Lean 4 上取得了最先进的结果，但该范式也带来了其独特的挑战。它需要在无法访问中间战术状态的情况下进行长视野序列预测，而未来的战术又依赖于这些隐藏的结果。在 Lean 的战术模式中，证明是通过一系列转换证明状态的战术构建的。这种顺序性引入了误差累积的风险 (Ross et al., 2011)，即单次误判就可能导致证明路径严重偏离正确方向。更具体地说，在生成长证明时，自回归模型可能会对中间战术状态产生错误认知。

为了在证明步骤生成中无缝集成中间战术状态，同时保持完整证明生成的简洁性与计算效率，我们在 DeepSeek-Prover-V1.5 中提出了一种统一的方法。该方法通过一种截断与恢复（truncate-and-resume）机制，结合了证明步骤生成与完整证明生成两种技术的优势。该过程从标准的完整证明生成开始，语言模型根据定理陈述前缀补全证明代码。随后，Lean 证明器对该代码进行验证。若证明正确且完整，则流程终止。若检测到错误，代码将在第一条错误信息处被截断，后续代码将被丢弃。成功生成的证明代码随后被用作生成下一证明片段的提示。为提高模型新补全内容的准确性，我们将 Lean 4 证明器的最新状态作为注释附加在提示末尾。值得注意的是，我们的方法并不局限于从最后成功应用的战术处恢复。我们将截断与恢复机制集成到蒙特卡洛树搜索 (MCTS; Coulom, 2006) 中，其中截断点由树搜索策略进行调度。此外，我们提出了一种新颖的无奖励（reward-free）探索算法用于 MCTS，以解决证明搜索中的奖励稀疏问题。我们为树搜索智能体赋予内在动机（*a.k.a.* 好奇心 (Schmidhuber, 2010)），以广泛探索战术状态空间。这些算法模块扩展了我们完整证明生成模型的功能，使其成为交互式定理证明的灵活工具，能够有效利用证明助手的反馈并生成多样化的解决方案候选。

1.1. 本文贡献

我们提出了一种用于开发基于语言模型的形式化数学证明器的综合框架，该框架集成了多个关键组件：大规模数学预训练、形式化数学语料构建与增强、基于证明助手反馈的在线强化学习，以及用于定理证明长期规划的树搜索方法。预训练模型、监督微调模型和强化学习模型，以及蒙

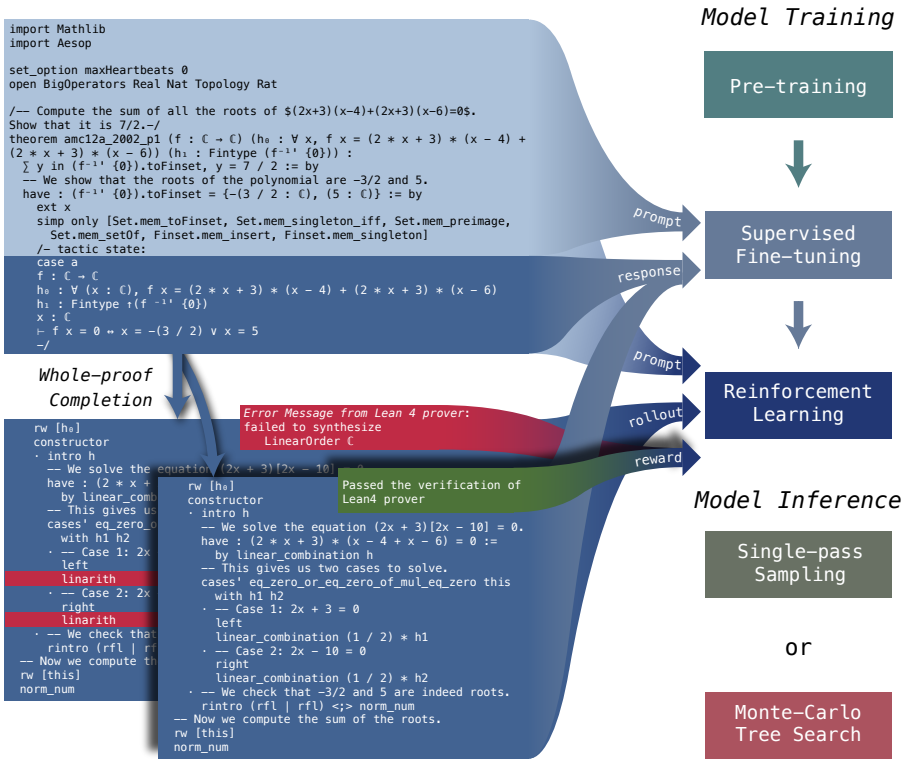


图 2 | **整体框架**。DeepSeek-Prover-V1.5 通过预训练、监督微调以及强化学习进行训练。在监督微调阶段，预训练模型接收一个以战术状态注释关键字结尾的不完整定理证明。模型被训练以预测该战术状态的内容（辅助目标）并补全后续证明步骤（主要目标）。在强化学习阶段，给定一个不完整定理证明以及来自 Lean 证明器的真实战术状态，我们展开微调后的模型以生成多个证明候选，随后由 Lean 证明器进行验证。这些候选的验证结果被用作二值（0-1）奖励，以进一步优化模型并增强其与验证系统形式化规范的对齐程度。在模型推理阶段，我们提供两种选择：单次采样与蒙特卡洛树搜索。

特卡洛树搜索算法的代码均已公开，以供进一步研究和应用。

- **预训练**：通过在高质量数学和代码数据上进行进一步预训练，我们增强了基础模型在形式化定理证明和数学推理方面的能力，重点关注 Lean、Isabelle 和 Metamath 等形式化语言。
- **监督微调**：我们通过实施两种数据增强技术改进了 Lean 4 代码补全数据集。首先，我们使用 DeepSeek-Coder V2 236B (Zhu et al., 2024) 为 Lean 4 代码添加自然语言思维链注释，使形式化定理证明与自然语言推理相一致。其次，我们在 Lean 4 证明代码中插入中间战术状态信息，使模型能够有效利用编译器反馈。随后，使用该数据集对预训练模型进行微调。
- **强化学习**：我们采用 GRPO 算法 (Shao et al., 2024) 对监督微调模型执行基于证明助手反馈的强化学习（RLPAF）。Lean 证明器的验证结果作为奖励监督信号，增强了模型与验证系统形式化规范的对齐程度。
- **蒙特卡洛树搜索**：我们通过引入一种新颖的抽象及其对应的搜索算法，推进了形式化定理证明中的树搜索方法。我们的截断与恢复机制作为一种状态-动作抽象，将树搜索过程无缝集成到完整证明生成框架中。我们提出了 RMaxTS，这是一种创新的蒙特卡洛树搜索算法，

利用 RMax (Brafman and Tennenholtz, 2002) 策略来解决稀疏奖励证明搜索问题中的探索挑战。通过分配内在奖励，该算法鼓励证明智能体生成多样化的规划路径，从而促进对证明空间的广泛探索。

1.2. 评估与指标总结

- **miniF2F**：在单次完整证明生成设置下，DeepSeek-Prover-V1.5 在 miniF2F 测试集上达到了 **60.2%** 的通过率，相较于 DeepSeek-Prover-V1 的 50.0% 实现了绝对 10.2 个百分点的显著提升。结合树搜索技术后，通过率进一步提升至新的最先进水平 **63.5%**。
- **ProofNet**：DeepSeek-Prover-V1.5 在 ProofNet 的单次完整证明生成设置中也表现出强劲性能，在验证集和测试集上的通过率分别为 **21.6%** 和 **23.7%**。集成树搜索技术后进一步提升了这些结果，在验证集和测试集上分别达到了新的最先进通过率 **25.4%** 和 **25.3%**。

2. 模型训练

2.1. 预训练

为了提升我们的语言模型在生成形式化证明和进行数学语言推理方面的能力，我们对基础模型 (Shao et al., 2024) 进行了进一步的预训练。该优化过程涉及在包含代码和自然语言数学内容的高质量数据集上进行训练。我们特别关注了在证明助手中广泛使用的形式化语言，例如 Lean、Isabelle 和 Metamath。我们将改进后的模型命名为 DeepSeek-Prover-V1.5-Base。

2.2. 监督微调

在本节中，我们探讨 DeepSeek-Prover-V1.5 监督微调（SFT）所涉及的方法与流程。具体而言，我们通过添加详细的解释性注释来增强来自 DeepSeek-Prover-V1 的证明数据集。该增强旨在改善自然语言描述与 Lean 4 代码之间的对齐，从而促进更好的形式化数学推理。此外，我们将中间战术状态信息作为辅助预测任务纳入其中，以支持蒙特卡洛树搜索过程中使用的截断与恢复机制。我们将得到的模型称为 DeepSeek-Prover-V1.5-SFT。

数据构建。 我们为监督微调构建了一个全面的 Lean 4 代码补全数据集。该数据集包含源自大量形式化定理的合成证明代码。这些定理来源于多个项目，例如标准的 Lean 4 数学库 Mathlib4 (Mathlib Community, 2020)、来自 DeepSeek-Prover-V1 (Xin et al., 2024) 和 Lean Workbook (Ying et al., 2024) 的合成定理，以及来自 miniF2F (Zheng et al., 2022) 和 ProofNet (Azerbayev et al., 2023) 基准的验证集。为了扩充形式化证明数据，我们采用了专家迭代过程 (Polu and Sutskever, 2020)。该过程包括使用语言模型生成证明、验证生成的证明数据、使用验证后的数据重新训练模型，然后利用优化后的模型生成更多的证明数据。在每次迭代之间，我们使用 DeepSeek-Coder V2 236B (Zhu et al., 2024) 将证明代码前的思维过程作为注释进行标注。最后，我们为蒙特卡洛树搜索的截断与恢复机制定制了这些数据（详见第 3.1节）。最终得到的证明数据集包含 9,645k 条序列。

思维增强的证明生成。 在 DeepSeek-Prover-V1 中，我们发现自然语言中的解题策略与 Lean 中的定理证明之间存在显著差距。在自然语言中，模型会生成详细的推导步骤来构建证明；而在 Lean 中，它们通常依赖一系列高层战术调用来暴力求解。这些高层战术虽然有效，但掩盖了其内部工作机制和结果，阻碍了模型利用结构化数学推理解决复杂证明目标的能力。为解决这一问题，我们开发了一种在生成定理证明代码前融入自然语言推理的方法。与 Lean-STaR (Lin et al., 2024) 类似（后者在每个证明步骤前执行独立的思维链推理 (Wei et al., 2022; Feng et al., 2023)），我们的方法将这种推理直接作为注释集成在证明代码中。我们使用 DeepSeek-Coder V2 236B (Zhu et al., 2024) 通过两种方式增强 DeepSeek-Prover-V1 中的现有数据：首先，在证明块的开头插入完整的自然语言解答；其次，交替插入对应 Lean 战术的具体自然语言步骤。使用这种数据格式训练模型，强制其在证明块开头提出完整的数学推理，并在每个战术前进行详细的步骤规划。该方法成功培养了新的行为模式，利用细致的数学思维来指导战术的生成。在训练数据中，使用两种不同的引导提示来区分证明代码补全的 CoT（思维链）模式与非 CoT 模式。两种模式下的输入输出示例见附录 A。

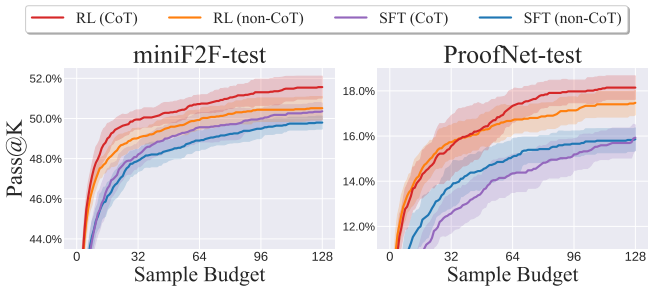
基于战术状态信息的提示增强。 为了实现蒙特卡洛树搜索的截断与恢复机制，我们需要从模型生成的代码中提取战术信息。我们利用 LeanDojo (Yang et al., 2023) 项目中的数据提取工具对 Lean REPL (Read-Eval-Print Loop; Leanprover Community, 2023) 进行了增强。这使得我们能够以三元组的形式提取战术信息，其中包括每个战术的位置，以及应用前后的战术状态。该信息有助于我们识别触发验证错误的具体战术代码（用于树搜索的扩展步骤，见第 3.2节）。对于生成的有效形式化证明中的每个战术，我们将验证器返回的战术状态作为注释"/- tactic state: ... -/" 插入。在训练期间，我们将"/- tactic state: " 之后的所有 token 作为响应来计算监督微调损失，而该注释之前的 token 则作为提示，不参与训练损失的计算。

训练设置。 我们基于预训练模型进行监督微调，训练数据量为 9B tokens，批次大小（batch size）为 2,048，学习率恒定为 1e-4。训练过程以 100 个预热步骤开始，以稳定学习动态。训练示例被随机拼接以形成序列，最大上下文长度为 4,096 tokens。

2.3. 基于证明助手反馈的强化学习

强化学习（RL）已被证明能有效提升监督微调语言模型的数学推理能力 (Shao et al., 2024)。为了进一步提升 DeepSeek-Prover-V1.5-SFT，我们引入了强化学习阶段，从而得到模型 DeepSeek-Prover-V1.5-RL。该阶段利用 RL 基于 Lean 4 证明器的验证反馈来增强模型性能。该 RL 过程的具体细节如下所述。

提示词。 在强化学习阶段，我们使用监督微调数据集中的一部分定理陈述作为训练提示词。我们选择那些 DeepSeek-Prover-V1.5-SFT 在多次尝试中生成正确证明的成功率适中的定理。这确保了模型仍有提升空间，同时仍能接收到正向反馈。经过筛选后，我们保留了约 4.5k 个唯一的定理陈述。每个定理前都添加了 CoT 和非 CoT 引导提示词，以增强模型在这两种模式下的证明生成能力。



模型	Pass@128	
	miniF2F-test	ProofNet-test
Base (3-shot)	29.7% ± 0.5%	9.7% ± 0.7%
SFT (非 CoT)	49.8% ± 0.3%	15.9% ± 0.5%
SFT (CoT)	50.4% ± 0.4%	15.9% ± 0.6%
RL (非 CoT)	50.5% ± 0.6%	17.5% ± 0.5%
RL (CoT)	51.6% ± 0.5%	18.2% ± 0.5%

图 3 | **不同训练阶段模型能力对比**。“CoT” 和 “非 CoT” 指使用两种引导提示词进行评估。阴影区域表示均值周围的标准差范围。符号 $\mu \pm \sigma$ 表示平均准确率 μ 和标准差 σ。

奖励机制。 在使用 RL 训练大语言模型时，通常由训练好的奖励模型提供反馈信号。相比之下，形式化定理证明得益于证明助手对生成证明的严格验证，具有显著优势。具体而言，如果生成的证明被验证为正确，则获得奖励 1，否则为 0。尽管这种二元奖励信号准确，但也较为稀疏，尤其对于监督微调模型而言具有挑战性的定理更是如此。为了缓解这种稀疏性，如上所述，我们选择了那些对监督微调模型具有挑战性但又可实现的训练提示词。

强化学习算法。 我们采用组相对策略优化 (GRPO; Shao et al., 2024) 作为我们的 RL 算法。与 PPO (Schulman et al., 2017) 相比，该算法在有效性和效率上均表现出显著优势，这主要归功于它无需训练额外的 Critic 模型。具体而言，GRPO 为每个定理提示词采样一组候选证明，并根据组内输出的相对奖励来优化模型。我们的提示词选择策略旨在使候选证明中很可能同时包含正确和错误的证明，这与 GRPO 的组相对特性高度契合，从而提升了训练过程。

训练设置。 我们基于 SFT 模型进行 RL 训练，该模型同时作为初始模型和用于施加 Kullback-Leibler (KL) 散度惩罚的参考模型。我们使用恒定的学习率 5e-6，KL 惩罚系数设置为 0.02。对于每个定理，我们采样一组 32 个候选证明，最大长度设置为 2,048。训练批次大小（batch size）配置为 512。

2.4. 评估

基准测试。 我们在以下基准测试上评估定理证明性能，以比较每个训练阶段后的模型能力：

- **MiniF2F** (Zheng et al., 2022) 侧重于高中水平练习和竞赛（如 AMC、AIME 和 IMO）的形式化解题能力，重点涵盖代数和数论。该基准包含 244 个验证问题和 244 个测试问题，最初基于 Lean 3，并根据 Yang (2023) 提供的版本手动转换为 Lean 4.9.0。
- **ProofNet** (Azerbayev et al., 2023) 评估本科数学水平的形式化定理证明能力。它包含来自广泛使用的本科教科书的 185 个验证问题和 186 个测试问题，涵盖实变与复变分析、线性代数、抽象代数和拓扑学。这些问题最初基于 Lean 3，并手动转换为 Lean 4.9.0。

提示词配置。 对于 DeepSeek-Prover-V1.5-Base 的每次证明尝试，我们独立地从验证集中采样三个证明示例来构建少样本提示词。对于 miniF2F 基准，我们使用 Yang (2023) 中的人工编写证明；而对于 ProofNet 基准，我们使用 DeepSeek-Prover-V1.5-RL 生成的正确证明作为少样本示例。对于 DeepSeek-Prover-V1.5-SFT 和 DeepSeek-Prover-V1.5-RL，我们采用两种类型的引导提示词：一种鼓励在每个证明步骤前进行思维链（CoT）推理，另一种则不鼓励（非 CoT）。详细示例见附录 A。

评估指标。 我们使用 pass@K 准确率指标来评估定理证明性能，该指标衡量模型在 K 次尝试内生成正确证明的成功率。每个模型部署在单张 A100-40G GPU 上，并利用 vLLM 框架 (Kwon et al., 2023) 进行样本生成。采样参数设置为温度（temperature）为 1，top-p 值为 0.95，最大 token 限制为 2,048。生成的证明随后使用 Lean 4 定理证明器进行验证。在此验证过程中，我们导入 Mathlib4 (Mathlib Community, 2020) 和 Aesop (Limperg and From, 2023) 以访问预定义的前提和战术。验证过程设有 300 秒的时间限制。

跨训练阶段的对比。 图 3 展示了各训练阶段在 miniF2F 和 ProofNet 数据集上的对比分析。我们的基础模型 DeepSeek-Prover-V1.5-Base 取得了显著的通过率，使用 3-shot 提示词解决了 miniF2F 基准测试集中近三分之一的问题。监督微调阶段得到的 DeepSeek-Prover-V1.5-SFT 显著优于基础模型，在 miniF2F 上的 Pass@128 准确率提升了约三分之二，在 ProofNet 上则翻了一番。随后的强化学习阶段进一步提升了模型性能，在所有 K 值下均提高了 Pass@K 准确率。与自然语言数学领域的发现（如 DeepSeekMath (Shao et al., 2024) 中报告的强化学习主要提升 TopK 正确响应）不同，我们观察到形式化定理证明基础能力的真正增强。这种提升不仅在较小的采样预算下显而易见，而且随着采样预算的增加保持稳定。这一结论在后续具有更大采样预算的蒙特卡洛树搜索实验中得到了进一步支持，详见第 4.2节。

CoT 与非 CoT 的对比。 我们对比了 DeepSeek-Prover-V1.5-SFT 和 DeepSeek-Prover-V1.5-RL 在非 CoT 和 CoT 生成模式下的性能。图 3 的结果表明，在大多数设置下，CoT 模式始终优于非 CoT 模式。具体而言，DeepSeek-Prover-V1.5-RL 借助这些增强的定理证明模式，在两个基准测试上均取得了更优的性能，在 miniF2F 上的平均准确率为 51.6%，在 ProofNet 上为 18.2%。CoT 模式中自然语言推理的整合显著增强了形式化证明编写的规划与执行。关于使用与不使用自然语言思维链的证明策略的详细对比，请参阅附录 A 中提供的示例。

3. 面向探索的蒙特卡洛树搜索

3.1. 战术级树抽象

为了在完整证明生成设置中实现树搜索方法，我们引入了一种证明树抽象来定义定制的状态和动作空间，并利用截断与恢复机制。大致遵循 Yao et al. (2023) 的范式，我们首先将不完整的证明分解为对应于各个证明步骤的树节点序列，然后利用这些树节点中存储的部分内容继续证明

生成过程。图 4 展示了从完整证明生成构建证明搜索树的过程。

截断：将证明分解为树节点。 我们在战术级别构建证明搜索树，其中每条树边代表战术状态的一次转换步骤。最初，我们将模型生成的完整证明提交给 Lean 证明器，将其解析为战术。随后，我们在最早的验证错误处截断证明，确保所有后续的战术代码都能成功应用，从而推动证明向目标定理推进。战术代码被分割为若干代码片段，每个片段包含一个有效的战术代码及其关联的思维链注释，对应于代表战术状态转换的单条树边。通过这种抽象，每个战术代码被转换为一系列树节点，形成从根节点到特定节点的路径。

恢复：从树节点生成证明。 在 Lean 4 中，不同的战术可能导致相同的战术状态，这意味着我们证明树中的每个节点可以对应多种实现相同结果的战术代码。为处理此情况，我们在每个节点存储一组这些等效的战术代码。当树搜索智能体扩展一个节点时，它会随机选择一个战术作为语言模型的提示。该提示包含以所选战术结尾的不完整证明代码，以及来自 Lean 证明器的战术状态信息（作为注释块）。微调后的模型（见第 2.2 节）已接受训练以识别并利用此格式，使用添加了战术状态注释的不完整代码来指导后续证明生成。

3.2. 基于蒙特卡洛树搜索的交互式定理证明

我们的证明搜索树基于标准的蒙特卡洛树搜索（MCTS）范式 (MCTS; Coulom, 2006; Browne et al., 2012) 构建，该范式迭代执行四个步骤：*选择*（Selection）、*扩展*（Expansion）、*模拟*（Simulation）和 *回溯*（Backpropagation）。我们将 *模拟*步骤整合至 *扩展*步骤中，因为我们的完整证明生成模型本质上会从扩展节点执行 rollout。算法工作流的详细设计如下。

选择。 选择步骤 *a.k.a.* 树策略，从根节点开始向下遍历，以识别一个有潜力的节点进行扩展。该算法步骤的目标是在探索（exploration）与利用（exploitation）之间取得平衡 (Kocsis and Szepesvári, 2006)。树节点 s 处的树策略通过从有效操作集合中选择使价值最大化的动作来计算：

$$TreePolicy(s) = \underset{a \in Children(s) \cup \{\oslash\}}{\arg\max} \; Q_{UCB}(s,a), \tag{1}$$

其中，动作 a 可以是移动到子节点（记为 $a \in Children(s)$），也可以是扩展当前节点 s（记为特殊符号 $a = \oslash$）。该方法采用了一种称为 *虚拟节点*（virtual node）的技术 (Wang et al., 2023)，为每个节点分配一个虚拟子节点，以表示选择当前节点 s 进行扩展。这使得树搜索智能体能够持续扩展非叶节点，因为其动作空间由生成模型支持，而该模型的输出范围无法通过固定次数的尝试来确定。在节点 s 上执行动作 a 的价值估计 $Q_{UCB}(s,a)$ 由两部分组成：

$$\forall a \in Children(s) \cup \{\oslash\}, \quad Q_{UCB}(s,a) = \underbrace{Q(s,a)}_{\text{Exploitation}} + \underbrace{UCB(s,a)}_{\text{Exploration}}, \tag{2}$$

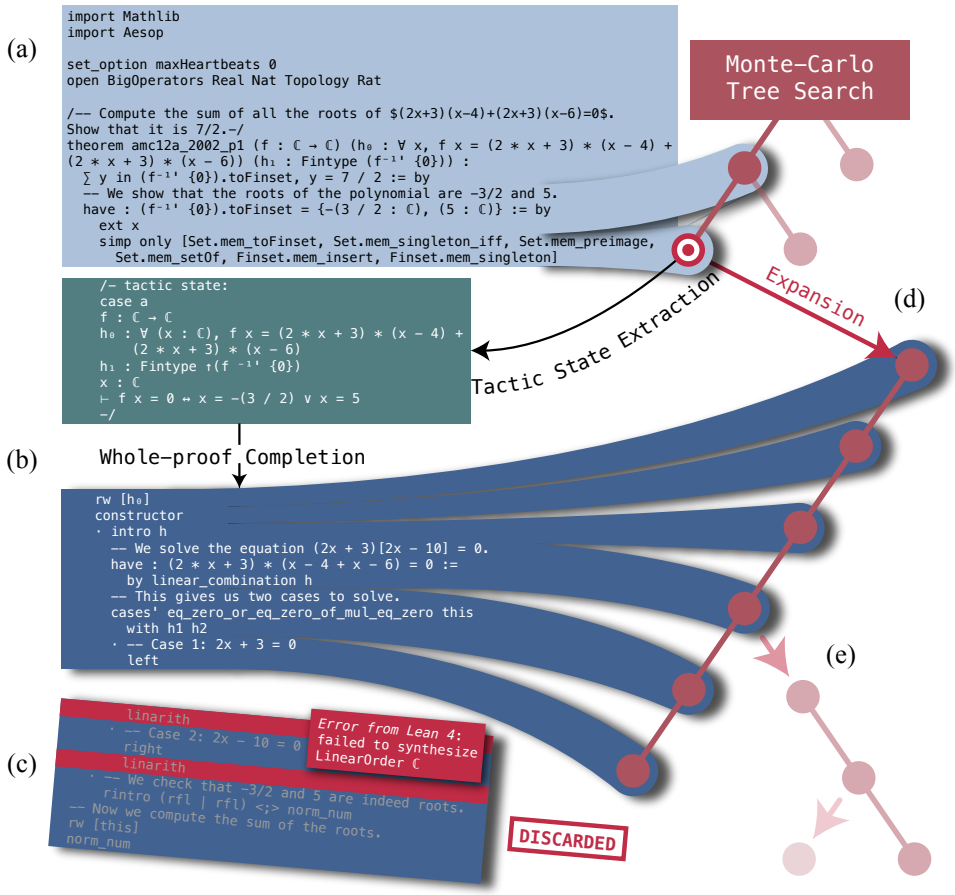


图 4 | **MCTS 扩展步骤中的截断与恢复机制**。 (a) 选定节点后，我们追溯其对应的不完整证明代码前缀，该前缀包含文件头、初始陈述以及来自祖先节点的成功应用的战术。(b) 随后，语言模型基于该前缀以及包含当前战术状态的注释块生成后续证明。(c) 组合后的证明代码（前缀与新生成代码）由 Lean 4 证明器进行验证。若未发现错误，树搜索过程终止。若检测到错误，我们在第一条错误信息处截断新生成的代码，丢弃后续代码，并将成功部分解析为战术。(d) 每个战术作为新节点添加到搜索树中，在所选节点下方扩展一条后代链。(e) 树更新完成后，下一次扩展迭代通过选择另一个候选节点开始，该节点不限于叶节点。此过程重复进行，直到找到正确的证明或采样预算耗尽。

其中，$Q(s,a)$ 表示基于选择历史得出的动作价值的样本估计，作为利用（exploitation）组件，用于从之前的尝试中检索高价值候选项。$UCB(s,a)$ 表示由上置信界（upper confidence bounds）计算的探索奖励 (UCB; Auer, 2002)，该奖励会随着状态-动作对 (s,a) 的重复执行而递减。更具体地说，$Q_{UCB}(s,a)$ 代表对 $Q(s,a)$ 的乐观估计，并以高概率作为其上界。关于节点价值和 UCB 奖励的详细设置讨论，我们将留至第 3.3 节。

扩展。 下一步是调用证明生成模型，以扩展由选择阶段选定的节点。通过恢复存储在待扩展节点上的不完整证明代码，我们执行完整证明生成以提出一系列后续战术，并将生成的证明提交给 Lean 证明器进行验证。这种证明完成尝试等效于在标准 MCTS 框架内执行单次模拟 rollout。当验证结果表明证明已完成时，搜索过程即可终止，此时已找到目标定理的新证明。否则，我们解析验证反馈，并将生成的证明截断至最早出现验证错误的断言处。剩余的战术将被转换为一

条节点路径，并合并到搜索树中（见图 4）。需要特别注意的是，由于我们采用完整证明生成设置——其输出是由一系列战术组成的完整证明，而不仅仅是下一个战术——我们的扩展过程可能在每次迭代中将一条树节点路径插入搜索树。这与专为竞技游戏设计的传统 MCTS 不同，后者通常每次迭代仅扩展一层子节点 (Silver et al., 2016, 2018; Schrittwieser et al., 2020)。

回溯。 每次树搜索迭代的最后阶段是沿从根节点到扩展节点的选择轨迹更新价值统计量，*i.e.*，更新与公式 (1) 中所述树策略相关的价值。令 $\tau = \{(root, s^{(1)}), (s^{(1)}, s^{(2)}), (s^{(2)}, s^{(3)}), \ldots, (s^{(|\tau|-1)} = s_t, \oslash)\}$ 表示第 t 次迭代的选择轨迹，该轨迹以 s_t 作为扩展节点结束。我们通过考虑最新的轨迹奖励 $R(\tau)$ 来更新所有 $(s, a) \in \tau$ 的 $Q_{UCB}(s, a)$（详见公式 (7)）。奖励的外在来源来自编译器反馈，具体而言，为已完成的证明分配奖励 $R_{\text{extrinsic}}(\tau) = 1$，为未解决的证明分配奖励 $R_{\text{extrinsic}}(\tau) = 0$。在第 3.3 节中，我们将引入一种内在奖励机制来增强奖励分配，从而提升智能体的探索激励。

3.3. 蒙特卡洛树搜索的内在奖励

在形式化定理证明的搜索问题中，外在奖励极其稀疏，*i.e.*，搜索智能体仅在证明完全解决时才能获得非零奖励。更具体地说，证明搜索过程形成了一棵树结构，只有少数叶子节点提供非零奖励，这与统计强化学习文献中著名的难探索案例 (Krishnamurthy et al., 2016) 相吻合。为了促进稀疏奖励序列决策中的探索，一种经典范式是构建内在奖励 (Schmidhuber, 2010)，鼓励智能体不仅优化外在奖励，还能获取关于交互环境的一般性信息 (Bellemare et al., 2016; Houthooft et al., 2016; Pathak et al., 2017; Burda et al., 2019)。在本节中，我们提出了一种由内在奖励驱动的探索算法——*应用于树搜索的 RMax*（RMaxTS），以在证明搜索问题中引入无奖励探索。

应用于 MCTS 的 RMax。 我们采用经典的探索机制 RMax (Brafman and Tennenholtz, 2002) 来为蒙特卡洛树搜索构建内在奖励。RMax 的核心思想是广泛探索状态空间。当智能体到达一个未见过的状态时，它会给自己分配最大量的奖励。在证明搜索的背景下，由于在证明完成前不提供任何外在奖励，我们的算法流程类似于 ZeroRMax (Jin et al., 2020)，其中智能体的探索完全由内在奖励驱动，*i.e.*，设定 $R(\tau) = R_{\text{intrinsic}}(\tau)$。树扩展步骤的内在奖励由是否向搜索树中添加了新节点决定，

$$R_{\text{intrinsic}}(\tau) = \mathbb{I}\left[\text{at least one new node is added to the search tree}\right], \tag{3}$$

其中 τ 表示最近的需要进行奖励分配以用于反向传播的选择轨迹。该探索策略优先扩展那些证明模型生成的战术会导致多样化战术状态的节点。由于多个 Lean 代码可能导致相同的中间状态转换，这种启发式方法有望减少冗余生成并提高样本效率。

针对非平稳奖励的 UCB。 蒙特卡洛树搜索中 UCB 探索奖励的常见设置是使用 UCB1 (Auer et al., 2002)：

$$Q_{UCB1}(s,a) = \frac{W(s,a)}{N(s,a)} + \sqrt{\frac{2\ln\sum_{a'} N(s,a')}{N(s,a)}}, \tag{4}$$

$$W(s,a) = \sum_{\tau\in\Gamma(s,a)} R(\tau), \tag{5}$$

$$N(s,a) = |\Gamma(s,a)|, \tag{6}$$

其中 $\Gamma(s,a) = \{\tau \mid (s,a) \in \tau\}$ 表示包含 (s,a) 作为中间选择步骤的树策略轨迹 τ 的列表。为了便于讨论，我们将列表 $\Gamma(s,a) = \{\tau_1, \tau_2, \cdots\}$ 组织为使得新收集的轨迹具有更大的下标索引。在本工作中，我们提出使用一种替代的 UCB 方法变体。需要注意的是，公式 (3) 中推导出的内在奖励是一个非平稳奖励信号，其期望值会随着探索的推进而衰减。这是因为随着搜索树通过精细的探索不断扩展，发现具有未见战术状态的新节点肯定会变得更加困难。为了解决非平稳性问题，我们考虑采用折扣上置信界 (DUCB; Garivier and Moulines, 2011)，该方法使用折扣因子 $\gamma \in (0,1)$ 来平滑地剔除那些过时的反馈记录：

$$Q_{DUCB}(s,a) = \frac{W_\gamma(s,a)}{N_\gamma(s,a)} + \sqrt{\frac{2\ln\sum_{a'} N_\gamma(s,a')}{N_\gamma(s,a)}}, \tag{7}$$

$$W_\gamma(s,a) = \sum_{t=1}^{N(s,a)} \gamma^{N(s,a)-t} R(\tau_t), \tag{8}$$

$$N_\gamma(s,a) = \sum_{t=0}^{N(s,a)-1} \gamma^t, \tag{9}$$

其中，新收到的反馈在价值估计中会被赋予更大的权重。在实践中，我们设定 $\gamma = 0.99$。需要注意的是，折扣因子 γ 在 DUCB 中的作用与其在无限视界 MDP 的值迭代中的作用不同。这里的折扣应用于树搜索迭代，而不是单个轨迹内的动作步长视界。

3.4. 蒙特卡洛树搜索的并行化

为了提高蒙特卡洛树搜索（MCTS）的效率，我们实现了Chaslot et al. (2008) 中描述的几种成熟的并行化技术。

- **根节点并行化：**我们在每个节点上部署 256 个 MCTS 运行器，每个 GPU 运行一个语言模型，证明生成的批处理大小为 512。Lean 证明器通过 REPL 调用，并在拥有数千个 CPU 核心的集群上执行，其中每个证明验证任务由一个独立进程处理，该进程在沙箱中创建和终止。语言模型的证明生成与 Lean 证明器的验证均异步处理。该设置允许 MCTS 运行器执行并发的树搜索操作，从而显著加速整个过程。
- **树并行化：**我们为每棵搜索树分配 32 个线程工作器，以并行化树迭代步骤。该方法有效地调度和平衡了证明生成与 Lean 验证的任务。每个线程工作器通过选择候选节点进行扩展、调用语言模型生成证明、使用 Lean 证明器验证生成的证明以及执行反向传播，迭代地执行树搜索循环。

- **虚拟损失：**为了鼓励并发的线程工作器选择不同的节点，我们为正在进行的迭代分配一个虚拟奖励 $R(\tau) = 0$。这涉及暂时反向传播奖励值 0，并在完成后将其更新为真实奖励。该策略促进了对不同扩展节点的探索，从而提高了整体搜索效率。

3.5. 与现有方法的比较

在本节中，我们将我们提出的证明树搜索方法与现有方法进行比较，该方法为完整证明生成引入了一种新颖的截断与恢复机制。目前，在形式数学证明搜索中使用语言模型的方法通常分为两大类策略：

- **多轮证明步骤生成**：该策略将证明过程分解为多个战术生成与验证的回合，通常遵循**树搜索**模式。它涉及一次生成并验证一个战术，重复该过程以处理下一个战术，直到没有剩余的证明目标为止。著名的例子包括 GPT-f (Polu and Sutskever, 2020; Polu et al., 2022)、Thor (Jiang et al., 2022)、ReProver (Yang et al., 2023)、Hypertree Proof Search (Lample et al., 2022) 以及 InternLM2-StepProver (Wu et al., 2024)。
- **单轮完整证明生成**：该方法尝试一次性生成并验证整个证明。如果证明不正确，模型将在下一次尝试中生成新的证明。此类方法包括 DSP (Jiang et al., 2022)、Subgoal-Prover Zhao et al. (2023)、LEGO-Prover (Wang et al., 2023)、Lyra (Zheng et al., 2023) 以及 miniCTX (Hu et al., 2024)。

我们的证明树搜索方法独特地桥接了这两种策略，提供了一种新颖的混合方法。它从完整证明生成开始，类似于单轮方法，但通过实现一种复杂的截断与恢复机制对此进行了扩展。该过程涉及将生成的证明截断至其成功的初始片段，将该片段解析为单独的战术，并从此处恢复树搜索。这种迭代过程有效地实现了蒙特卡洛树搜索，将单轮完整证明生成与多轮证明步骤生成无缝集成。因此，我们可以训练一个具有几乎相同目标的单一模型，以同时支持这两种策略。我们的实验结果表明，这种统一的方法在两种设置下均取得了优越的性能。通过结合现有方法的优势并引入创新技术，我们的方法为形式数学证明搜索提供了一种更通用且有效的解决方案，有望为该领域的未来进步铺平道路。

4. 实验结果

在本节中，我们使用两个不同的基准测试来评估 DeepSeek-Prover-V1.5 的定理证明能力：涵盖高中水平练习和竞赛题的 miniF2F，以及涉及本科水平定理的 ProofNet。我们展示了完整证明生成和蒙特卡洛树搜索方法的结果，并采用与第 2.4 节相同的训练模型和推理配置，以确保一致性。

4.1. 主要结果

我们将 DeepSeek-Prover-V1.5 与以往最先进的语言模型进行了对比分析，以突出其性能与进展。

- **通用模型：** **GPT-3.5** 和 **GPT-4** (OpenAI, 2023) 是由 OpenAI 开发的先进生成式 AI 模型，以其在包括代码生成在内的多种任务中的有效性而闻名。尽管它们并非专为定理证明设计，但其庞大的参数规模赋予了显著的能力。在形式化定理证明中，这些模型的评估得益于 **COPRA** (Thakur et al., 2023)，这是一种上下文学习智能体，利用这些大语言模型来提出策略应用。此外，我们还考察了 **Llemma** (Azerbayev et al., 2024)，这是一系列在大量通用数学语料库上训练的语言模型，常被用作形式化定理证明的基础模型。
- **形式化数学专用模型：** **GPT-f** (Polu and Sutskever, 2020; Polu et al., 2022) 代表了将 Transformers (Vaswani et al., 2017) 应用于定理证明任务的证明步骤生成的初步尝试，并利用最佳优先搜索模块来构建完整证明。后续的进展包括 **ReProver** (Yang et al., 2023)、**LLMStep** (Welleck and Saha, 2023) 和 **Lean-STaR** (Lin et al., 2024)。**Hypertree Proof Search** (Lample et al., 2022) 探索了在基于 Lean 的形式化定理证明中使用蒙特卡洛树搜索的方法。同期工作 **InternLM2-Math** (Ying et al., 2024) 和 **InternLM2-StepProver** (Wu et al., 2024) 也展现了出色的性能。

评估指标。 我们使用 pass@K 准确率指标将 DeepSeek-Prover-V1.5 的性能与最先进模型进行比较，该指标评估模型在 K 次尝试内生成正确证明的能力。为了在不同生成方案之间对齐计算预算，我们按照以下规则展示采样预算 K。

- 对于单次采样方法，我们将采样预算 K 定义为生成的证明总数，并将较大的 K 值进行因式分解，以便于与树搜索方法进行比较。
- 对于最佳优先搜索方法，遵循 Azerbayev et al. (2024) 的符号表示，我们给出 $K = N \times S \times T$，其中 N 表示最佳优先搜索的尝试次数，S 表示每次扩展生成的策略数量，T 表示扩展迭代次数。
- 对于树搜索方法，*e.g.*, RMaxTS 和 HTPS (Lample et al., 2022)，我们给出 $K = N \times T$，其中 N 表示树搜索的尝试次数，T 表示树扩展中调用的模型生成次数。

miniF2F 数据集上的结果。 表 1 提供了 miniF2F-test 数据集上各种定理证明方法的对比分析。在单次完整证明生成设置下，DeepSeek-Prover-V1.5-RL 取得了 60.2% 的最高通过率，较 DeepSeek-Prover-V1 的 50.0% 显著提升了 10.2 个百分点。在采样预算限制为 128 次尝试的情况下，DeepSeek-Prover-V1.5-RL 证明了 51.6% 的问题，显著优于其他完整证明生成方法，且与领先的树搜索方法相当。在树搜索方法类别中，DeepSeek-Prover-V1.5-RL + RMaxTS 以 62.7% 的通过率位居榜首，确立了新的最先进水平，并与现有方法拉开了显著差距。值得注意的是，DeepSeek-Prover-V1.5-RL 仅需 3200 次完整证明生成采样即可达到 54.9% 的通过率，超越了此前 InternLM2-StepProver 的最先进结果（后者执行了 64×3200 次树搜索才达到 54.5%）。

ProofNet 数据集上的结果。 表 2 展示了 ProofNet 数据集上各种定理证明方法的对比分析。在单次完整证明生成设置下，以及结合 RMaxTS 增强后，DeepSeek-Prover-V1.5-RL 在整体 ProofNet 数据集上分别取得了 22.6% 和 25.3% 的通过率。这些结果超越了现有的最先进方法

方法	采样预算	miniF2F-test
单次完整证明生成方法		
TheoremLlama [49]	128	33.6%
DeepSeek-Prover-V1 [53]	128	46.1% ± 0.5%
	16×4096	50.0%
DeepSeek-Prover-V1.5-Base	128	29.7% ± 0.5%
	3200	39.2%
	6400	42.2%
DeepSeek-Prover-V1.5-SFT	32	48.2% ± 0.6%
	64	49.6% ± 0.7%
	128	50.4% ± 0.4%
	3200	53.3% ± 0.5%
	4×6400	55.8% ± 0.7%
	16×6400	57.4%
DeepSeek-Prover-V1.5-RL	32	50.0% ± 0.5%
	64	50.7% ± 0.4%
	128	51.6% ± 0.5%
	3200	54.9% ± 0.7%
	4×6400	58.4% ± 0.6%
	16×6400	**60.2**%
树搜索方法		
COPRA (Code Llama) [45]	1×500	5.7%
COPRA (GPT-3.5) [45]	1×60	9.0%
COPRA (GPT-4) [45]	1×60	26.6%
Llemma-7B [5]	$1 \times 32 \times 100$	26.2%
Llemma-34B [5]	$1 \times 32 \times 100$	25.8%
ReProver [55]	-	26.5%
LLMStep [51]	$1 \times 32 \times 100$	27.9%
GPT-f [35]	$64 \times 8 \times 512$	36.6%
Hypertree Proof Search [23]	64×5000	41.0%
Lean-STaR [26]	$64 \times 1 \times 50$	46.3%
InternLM2-Math-7B [58]	$1 \times 32 \times 100$	30.3%
InternLM2-Math-Plus-7B [58]	$1 \times 32 \times 100$	43.4%
InternLM2-StepProver [52]	$1 \times 32 \times 100$	48.8%
	$64 \times 32 \times 100$	54.5%
DeepSeek-Prover-V1.5-SFT + RMaxTS	1×3200	53.5% ± 0.4%
	4×6400	56.3% ± 0.3%
	16×6400	59.0%
	$32 \times 6400^{\dagger}$	60.2%
DeepSeek-Prover-V1.5-RL + RMaxTS	1×3200	55.0% ± 0.7%
	4×6400	59.6% ± 0.6%
	16×6400	**62.7**%
	$32 \times 6400^{\dagger}$	**63.5**%

表 1 | 在 miniF2F-test 数据集上与最先进方法的对比。符号 $\mu \pm \sigma$ 表示平均准确率 μ 和标准差 σ。除非另有说明，DeepSeek-Prover-V1.5-Base 的结果基于 3-shot 提示，而 DeepSeek-Prover-V1.5-SFT 和 RL 采用 CoT 模式提示。符号 † 表示使用两种引导提示的混合策略的性能（详见第 4.2节）。

方法	采样预算	ProofNet 验证集 ‡	ProofNet 测试集	全部
单次完整证明生成方法				
DeepSeek-Prover-V1.5-Base	128	6.6% ± 0.9%	9.7% ± 0.7%	7.5% ± 0.7%
	3200	10.8%	15.6%	13.2%
DeepSeek-Prover-V1.5-SFT	128	19.9% ± 0.4%	15.9% ± 0.6%	17.9% ± 0.3%
	3200	20.7% ± 0.7%	21.0% ± 0.9%	20.9% ± 0.6%
	4 × 6400	22.2%	23.7%	22.9%
DeepSeek-Prover-V1.5-RL	128	20.1% ± 0.5%	18.2% ± 0.5%	19.1% ± 0.4%
	3200	21.4% ± 0.3%	22.0% ± 0.5%	21.7% ± 0.4%
	4 × 6400	21.6%	23.7%	22.6%
树搜索方法				
ReProver [55]	-	-	-	13.8%
InternLM2-StepProver [52]	1 × 32 × 100	-	-	18.1%
DeepSeek-Prover-V1.5-SFT + RMaxTS	1 × 3200	22.2% ± 0.7%	21.6% ± 0.2%	21.9% ± 0.4%
	4 × 6400	23.8%	**25.8%**	24.8%
DeepSeek-Prover-V1.5-RL + RMaxTS	1 × 3200	22.0% ± 0.3%	21.5% ± 0.8%	21.8% ± 0.4%
	4 × 6400	25.4%	**25.3%**	25.3%

表 2 | 在 ProofNet 数据集上与最先进方法的对比。‡ 注意，ProofNet 的验证集用于在监督微调中进行专家迭代。

ReProver (13.8%) 和 InternLM2-StepProver (18.1%)。当完整证明生成尝试次数限制为 3200 次时, DeepSeek-Prover-V1.5 也证明了 21.7% 的定理, 较此前的最先进方法 InternLM2-StepProver 提升了 3.6%。

4.2. 重新审视训练策略在大规模采样中的有效性

我们在大规模采样设置下重新审视了多个训练模块的效果，重点关注单次完整证明生成和蒙特卡洛树搜索。结果表明，第 2.4 节中提出的观察和发现能够推广到大规模采样的场景中。

强化学习的普遍提升。 为了支持“基于验证反馈的在线强化学习能够普遍提升模型能力”这一观点, 我们在大采样预算下将最终模型与仅使用监督微调 (SFT) 的版本进行了对比。对比结果以两列的形式展示在表 3 中。DeepSeek-Prover-V1.5-RL 在所有生成设置下均持续优于 SFT 模型, 无论是否应用思维链 (CoT) 策略。结果还表明, 在线强化学习带来的提升与通过 RMaxTS 获得的提升是正交的, 两者可以进一步结合以提升性能。通过结合 CoT 提示和 RMaxTS, DeepSeek-Prover-V1.5-RL 在 miniF2F-test 上达到了 62.7% 的通过率。这一性能较 SFT 模型显著提升了 3.7%，凸显了强化学习在提升证明补全模型整体有效性方面的关键作用。

CoT、非 CoT 与混合策略。 我们在 miniF2F-test 数据集上对比了两种生成模式, *i.e.*, 非 CoT 和 CoT 的性能。表 3 中的结果表明，随着采样预算的增加，CoT 相对于非 CoT 模式的优势被进一步放大。这表明, 引入自然语言思维链可以多样化定理证明的规划路径, 从而可能带来更广泛的推理策略和更具创新性的解决方案。结果还显示，这两种模式在不同问题上具有互补优势。

	提示模式	采样预算	DeepSeek-Prover-V1.5 SFT	DeepSeek-Prover-V1.5 RL
单次生成	non-CoT	4×6400	$54.7\%\pm0.4\%$	$56.5\%\pm0.5\%$
		16×6400	56.1%	57.4%
	CoT	4×6400	$55.8\%\pm0.7\%$	$58.4\%\pm0.5\%$
		16×6400	57.4%	60.2%
	non-CoT 与 CoT	$(2+2)\times6400$	$56.1\%\pm0.8\%$	$58.3\%\pm0.6\%$
		$(8+8)\times6400$	58.2%	60.7%
		$(16+16)\times6400$	58.6%	61.1%
RMaxTS	non-CoT	4×6400	$55.7\%\pm0.6\%$	$58.4\%\pm0.6\%$
		16×6400	57.8%	59.4%
	CoT	4×6400	$56.3\%\pm0.3\%$	$59.6\%\pm0.6\%$
		16×6400	59.0%	62.7%
	non-CoT 与 CoT	$(2+2)\times6400$	$56.1\%\pm0.8\%$	$60.0\%\pm0.8\%$
		$(8+8)\times6400$	59.0%	63.1%
		$(16+16)\times6400$	60.2%	**63.5%**

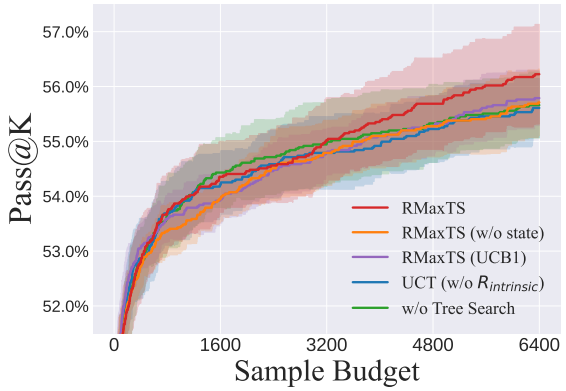
表 3 | 一项大规模消融实验，旨在探究几种算法设计对模型训练的有效性。结果在 miniF2F-test 数据集上进行评估。

在 CoT 模式下，模型的定理证明策略在数学思维上更加系统和主动；而在非 CoT 模式下，模型能够高效利用 Lean 高级战术来解决 Lean 自动化机制内可处理的计算问题。为了利用这些优势，我们采用了一种混合策略（在表 3 中标记为 non-CoT & CoT），将一半的采样预算分配给 CoT 模式，其余分配给非 CoT 模式。这两种引导提示的简单组合在进一步提升我们证明补全模型性能方面展现出巨大潜力，在 miniF2F-test 上达到了 63.5% 的通过率。在附录 B 中，我们展示了说明这两种生成模式不同优势的示例问题。

4.3. RMaxTS 的消融实验

内在奖励与折扣 UCB。 我们研究了 RMaxTS 的两个核心组件的有效性，*i.e.*，公式 (3) 中定义的内在奖励和公式 (7) 中给出的折扣上置信界（discounted UCB）。我们首先以一个未使用内在奖励的标准 UCT 算法 (Kocsis and Szepesvári, 2006) 基线开始，在该基线中，探索完全由 UCB 奖励驱动。需要注意的是，由于该基线未提供任何非零奖励，所有 UCB 公式的变体都变得等价，因为节点选择仅由访问次数决定。图 5 中的实验结果表明，在没有内在奖励的情况下，UCT（无 $R_{\text{intrinsic}}$）的性能退化到与非搜索方法相当的水平。此外，我们考虑了使用标准 UCB1（参见公式 (4)）替代折扣 UCB 的 RMaxTS，记为 RMaxTS (DUCB $\rightarrow$ UCB1)。结果表明，带有 UCB1 奖励的 RMaxTS 性能也较为有限，与 UCT（无 $R_{\text{intrinsic}}$）相当。这是因为 UCB1 旨在通过充分探索来保证渐近性能 (Auer et al., 2002)，其前提是样本量足够大。相比之下，折扣 UCB 能够加速非平稳内在奖励的价值传播，防止 $R_{\text{intrinsic}}$ 的引导作用被访问次数主导。这证明了折扣 UCB 机制是对内在奖励驱动探索的关键补充。

战术状态信息的引导作用。 在扩展树节点时，我们将中间战术状态信息作为注释块拼接到不完整代码中，以引导证明补全。借助提供的辅助信息，证明补全模型能够增强其对战术状态的内



	采样预算	miniF2F-test
单次生成	4×6400 16×6400	$58.4\% \pm 0.5\%$ 60.2%
UCT （无 $R_{\text{intrinsic}}$）	4×6400 16×6400	$58.2\% \pm 0.3\%$ 61.1%
RMaxTS （DUCB → UCB1）	4×6400 16×6400	$58.6\% \pm 0.3\%$ 60.7%
RMaxTS （无战术状态）	4×6400 16×6400	$58.4\% \pm 0.3\%$ 61.1%
RMaxTS	4×6400 16×6400	$59.6\% \pm 0.6\%$ 62.7%

图 5 | 一项考察 RMaxTS 算法设计的模块化消融实验。实验在 miniF2F-test 数据集上进行，使用处于 CoT 模式的 DeepSeek-Prover-V1.5-RL。左面板展示了 6400 次生成样本内的 Pass@K 准确率曲线。右面板展示了更大样本量下的结果。

部表征，为长视野规划提供中间引导。为了证明这一优势，我们展示了 RMaxTS 直接从原始不完整代码进行代码补全而不访问战术状态信息的实验，在图 5 中记为 RMaxTS (without tactic state)。结果表明，在没有战术状态信息的情况下，应用树搜索带来的性能提升变得较为有限，尤其是在处理需要大量样本的难题时。这凸显了编译器信息的集成是树搜索算法的重要组成部分，能够提升其整体有效性和样本效率。

5. 结论、局限性与未来工作

我们提出了 DeepSeek-Prover-V1.5，这是一个拥有 70 亿参数的语言模型，在 Lean 4 的形式化定理证明任务上超越了所有开源模型。DeepSeek-Prover-V1.5 以 DeepSeek-Prover-V1.5-Base 为初始化基础，后者使用专门针对形式化数学推理的语料库对 DeepSeekMath-Base 7B 进行了预训练扩展。我们在一个全面的 Lean 4 代码补全数据集上进行了监督微调，该数据集涵盖了来自各个数学领域的广泛形式化定理。随后，我们采用 GRPO 通过在线强化学习来提升完整证明的生成能力。在开发 DeepSeek-Prover-V1.5 模型的基础上，我们引入了 RMaxTS（一种蒙特卡洛树搜索的变体），通过具有广泛探索性的大规模搜索来提升问题解决能力。这些组件构成了一个用于训练基于大语言模型的证明助手的完整流水线，使 DeepSeek-Prover-V1.5 相较于 DeepSeek-Prover-V1 取得了显著的提升。

DeepSeek-Prover-V1.5 的框架旨在为形式化定理证明建立一个类似 AlphaZero 的流水线。专家迭代与合成数据的使用映射了强化学习的核心试错循环，其中编译器预言机充当世界模型以提供环境监督。在强化学习范式下，集成的树搜索模块已被证明在推动各领域实现超人类表现方面具有极高的有效性 (Silver et al., 2016; Fawzi et al., 2022; Lutz et al., 2023)。在开发 DeepSeek-Prover-V1.5 的过程中，我们侧重于强化学习的探索方面，引入 RMaxTS 以多样化证明步骤的生成。然而，利用方面，特别是证明搜索问题，仍有待探索。一个极具前景的未来方向是训练一个评论家模型来评估不完整的证明并剪枝搜索分支。此类针对部分证明的评论家模型将隐式地执行时序信用分配 (Sutton, 1984)，将证明级别的反馈分解为逐步的价值差异 (Arjona-

Medina et al., 2019)。开发用于评估长规划路径并提供引导奖励的评论家模型是一个至关重要且充满挑战的问题 (Ng and Russell, 2000; Sorg et al., 2010)，值得进一步研究。

最后，近期的工作已不再局限于证明单个定理，而是转向处理复杂的多定理 Lean 文件中的现实世界理论证明 (Hu et al., 2024)。这一转变是我们完整证明生成方法的自然延伸。我们的观察表明，当前模型已经具备了对文件级上下文的一定理解能力。展望未来，我们将专注于增强这一方面，利用我们在语言模型方面的进展来支持前沿的 Lean 数学形式化开发者。

参考文献

[1] J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter. Rudder: Return decomposition for delayed rewards. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, pages 13566–13577, 2019.

[2] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. Journal of Machine Learning Research, 3(Nov):397–422, 2002.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. Machine learning, 47:235–256, 2002.

[4] Z. Azerbayev, B. Piotrowski, H. Schoelkopf, E. W. Ayers, D. Radev, and J. Avigad. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. arXiv preprint arXiv:2302.12433, 2023.

[5] Z. Azerbayev, H. Schoelkopf, K. Paster, M. Dos Santos, S. M. McAleer, A. Q. Jiang, J. Deng, S. Biderman, and S. Welleck. Llemma: An open language model for mathematics. In The Twelfth International Conference on Learning Representations, 2024.

[6] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In Proceedings of the 30th International Conference on Neural Information Processing Systems, pages 1479–1487, 2016.

[7] R. I. Brafman and M. Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. Journal of Machine Learning Research, 3(Oct):213–231, 2002.

[8] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in games, 4(1):1–43, 2012.

[9] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. In Seventh International Conference on Learning Representations, pages 1–17, 2019.

[10] G. M. B. Chaslot, M. H. Winands, and H. J. van Den Herik. Parallel monte-carlo tree search. In Computers and Games: 6th International Conference, CG 2008, Beijing, China, September 29-October 1, 2008. Proceedings 6, pages 60–71. Springer, 2008.

[11] R. Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In International conference on computers and games, pages 72–83. Springer, 2006.

[12] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R Ruiz, J. Schrittwieser, G. Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. Nature, 610(7930):47–53, 2022.

[13] G. Feng, B. Zhang, Y. Gu, H. Ye, D. He, and L. Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. In Thirty-seventh Conference on Neural Information Processing Systems, volume 36, 2023.

[14] A. Garivier and E. Moulines. On upper-confidence bound policies for switching bandit problems. In International conference on algorithmic learning theory, pages 174–188. Springer, 2011.

[15] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. Vime: variational information maximizing exploration. In Proceedings of the 30th International Conference on Neural Information Processing Systems, pages 1117–1125, 2016.

[16] J. Hu, T. Zhu, and S. Welleck. minictx: Neural theorem proving with (long-)contexts, 2024.

[17] A. Q. Jiang, W. Li, S. Tworkowski, K. Czechowski, T. Odrzygóźdź, P. Miłoś, Y. Wu, and M. Jamnik. Thor: wielding hammers to integrate language models and automated theorem provers. In Proceedings of the 36th International Conference on Neural Information Processing Systems, pages 8360–8373, 2022.

[18] A. Q. Jiang, S. Welleck, J. P. Zhou, T. Lacroix, J. Liu, W. Li, M. Jamnik, G. Lample, and Y. Wu. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In The Eleventh International Conference on Learning Representations, 2022.

[19] C. Jin, A. Krishnamurthy, M. Simchowitz, and T. Yu. Reward-free exploration for reinforcement learning. In International Conference on Machine Learning, pages 4870–4879. PMLR, 2020.

[20] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In European conference on machine learning, pages 282–293. Springer, 2006.

[21] A. Krishnamurthy, A. Agarwal, and J. Langford. PAC reinforcement learning with rich observations. In Proceedings of the 30th International Conference on Neural Information Processing Systems, pages 1848–1856, 2016.

[22] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with page-dattention. In Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles, 2023.

[23] G. Lample, M.-A. Lachaux, T. Lavril, X. Martinet, A. Hayat, G. Ebner, A. Rodriguez, and T. Lacroix. Hypertree proof search for neural theorem proving. In Proceedings of the 36th International Conference on Neural Information Processing Systems, pages 26337–26349, 2022.

[24] Leanprover Community. A read-eval-print-loop for Lean 4. `https://github.com/leanprover-community/repl`, 2023.

[25] J. Limperg and A. H. From. Aesop: White-box best-first proof search for lean. In Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs, pages 253–266, 2023.

[26] H. Lin, Z. Sun, Y. Yang, and S. Welleck. Lean-star: Learning to interleave thinking and proving. arXiv preprint arXiv:2407.10040, 2024.

[27] I. D. Lutz, S. Wang, C. Norn, A. Courbet, A. J. Borst, Y. T. Zhao, A. Dosey, L. Cao, J. Xu, E. M. Leaf, et al. Top-down design of protein architectures with reinforcement learning. Science, 380(6642):266–273, 2023.

[28] Mathlib Community. The Lean mathematical library. In Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, page 367–381. Association for Computing Machinery, 2020.

[29] L. d. Moura and S. Ullrich. The lean 4 theorem prover and programming language. In Automated Deduction–CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28, pages 625–635. Springer, 2021.

[30] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In Proceedings of the Seventeenth International Conference on Machine Learning, pages 663–670, 2000.

[31] OpenAI. Gpt-4 technical report. arXiv preprint arXiv:2303.08774, 2023.

[32] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In International conference on machine learning, pages 2778–2787. PMLR, 2017.

[33] L. C. Paulson. Isabelle a Generic Theorem Prover. Springer Verlag, 1994.

[34] S. Polu and I. Sutskever. Generative language modeling for automated theorem proving. arXiv preprint arXiv:2009.03393, 2020.

[35] S. Polu, J. M. Han, K. Zheng, M. Baksys, I. Babuschkin, and I. Sutskever. Formal mathematics statement curriculum learning. arXiv preprint arXiv:2202.01344, 2022.

[36] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Proceedings of the fourteenth international conference on artificial intelligence and statistics, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[37] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). IEEE transactions on autonomous mental development, 2(3):230–247, 2010.

[38] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. Nature, 588(7839):604–609, 2020.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[40] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. Li, Y. Wu, and D. Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024.

[41] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. nature, 529(7587):484–489, 2016.

[42] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science, 362(6419):1140–1144, 2018.

[43] J. Sorg, S. Singh, and R. L. Lewis. Reward design via online gradient ascent. In Proceedings of the 23rd International Conference on Neural Information Processing Systems-Volume 2, pages 2190–2198, 2010.

[44] R. S. Sutton. Temporal credit assignment in reinforcement learning. Phd thesis, University of Massachusetts, 1984.

[45] A. Thakur, Y. Wen, and S. Chaudhuri. A language-agent approach to formal theorem-proving. In The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS, 2023.

[46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[47] H. Wang, H. Xin, C. Zheng, Z. Liu, Q. Cao, Y. Huang, J. Xiong, H. Shi, E. Xie, J. Yin, et al. Lego-prover: Neural theorem proving with growing libraries. In The Twelfth International Conference on Learning Representations, 2023.

[48] H. Wang, Y. Yuan, Z. Liu, J. Shen, Y. Yin, J. Xiong, E. Xie, H. Shi, Y. Li, L. Li, et al. DT-solver: Automated theorem proving with dynamic-tree sampling guided by proof-level value function. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12632–12646, 2023.

[49] R. Wang, J. Zhang, Y. Jia, R. Pan, S. Diao, R. Pi, and T. Zhang. Theoremllama: Transforming general-purpose llms into lean4 experts. arXiv preprint arXiv:2407.03203, 2024.

[50] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Proceedings of the 36th International Conference on Neural Information Processing Systems, pages 24824–24837, 2022.

[51] S. Welleck and R. Saha. llmstep: Llm proofstep suggestions in lean. In The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23, 2023.

[52] Z. Wu, J. Wang, D. Lin, and K. Chen. Lean-github: Compiling github lean repositories for a versatile lean prover. arXiv preprint arXiv:2407.17227, 2024.

[53] H. Xin, D. Guo, Z. Shao, Z. Ren, Q. Zhu, B. Liu, C. Ruan, W. Li, and X. Liang. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. arXiv preprint arXiv:2405.14333, 2024.

[54] K. Yang. minif2f-lean4. `https://github.com/yangky11/miniF2F-lean4`, 2023.

[55] K. Yang, A. M. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. Prenger, and A. Anandkumar. Leandojo: theorem proving with retrieval-augmented language models. In Proceedings of the 37th International Conference on Neural Information Processing Systems, pages 21573–21612, 2023.

[56] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: deliberate problem solving with large language models. In Proceedings of the 37th International Conference on Neural Information Processing Systems, pages 11809–11822, 2023.

[57] H. Ying, Z. Wu, Y. Geng, J. Wang, D. Lin, and K. Chen. Lean workbook: A large-scale lean problem set formalized from natural language math problems. arXiv preprint arXiv:2406.03847, 2024.

[58] H. Ying, S. Zhang, L. Li, Z. Zhou, Y. Shao, Z. Fei, Y. Ma, J. Hong, K. Liu, Z. Wang, et al. Internlm-math: Open math large language models toward verifiable reasoning. arXiv preprint arXiv:2402.06332, 2024.

[59] X. Zhao, W. Li, and L. Kong. Decomposing the enigma: Subgoal-based demonstration learning for formal theorem proving. arXiv preprint arXiv:2305.16366, 2023.

[60] C. Zheng, H. Wang, E. Xie, Z. Liu, J. Sun, H. Xin, J. Shen, Z. Li, and Y. Li. Lyra: Orchestrating dual correction in automated theorem proving. arXiv preprint arXiv:2309.15806, 2023.

[61] K. Zheng, J. M. Han, and S. Polu. miniF2F: a cross-system benchmark for formal olympiad-level mathematics. In International Conference on Learning Representations, 2022.

[62] Q. Zhu, D. Guo, Z. Shao, D. Yang, P. Wang, R. Xu, Y. Wu, Y. Li, H. Gao, S. Ma, et al. DeepSeek-Coder-V2: Breaking the barrier of closed-source models in code intelligence. arXiv preprint arXiv:2406.11931, 2024.

A. 证明补全中非思维链与思维链提示的示例

在本节中，我们展示了针对 `aime_1983_p9`（来自 miniF2F 基准测试 (Zheng et al., 2022) 的一道题目）的非思维链与思维链提示示例及其生成的证明。为便于阅读，文中使用了行内 LaTeX 表达式。首先，以下是非思维链提示模式下完整证明生成的示例：

输入：

````
Complete the following Lean 4 code:

```lean4
import Mathlib
import Aesop

set_option maxHeartbeats 0

open BigOperators Real Nat Topology Rat

/-- Show that (9*x^2*sin^2 x+4)/(x*sin x) ≥ 12 for 0 < x < π.-/
theorem aime_1983_p9 (x : ) (h : 0 < x  x < Real.pi) :
  12  (9 * (x ^ 2 * Real.sin x ^ 2) + 4) / (x * Real.sin x) := by
````

输出：

````
  rw [le_div_iff (mul_pos h.1 (Real.sin_pos_of_pos_of_lt_pi h.1 h.2))]
  nlinarith [mul_self_nonneg (3 * x * Real.sin x - 2)]
```
````

接下来，以下是思维链提示模式下完整证明生成的示例。该证明是通过带有截断与恢复机制的蒙特卡洛树搜索找到的（详见第 3 节）。模型接收到的提示包含一段正在进行的证明代码，末尾附有一个注释块，其中记录了 Lean 4 证明器返回的中间战术状态。后续代码的生成则在该带注释的代码前缀引导下完成。

输入：

````
Complete the following Lean 4 code with explanatory comments preceding each line of
    code:

```lean4
import Mathlib
import Aesop

set_option maxHeartbeats 0

open BigOperators Real Nat Topology Rat
````
````

```
/-- Show that (9x^2 sin^2 x+4)/(x sin x) ≥ 12 for 0 < x < π.-/
theorem aime_1983_p9 (x : ℝ) (h : 0 < x ∧ x < Real.pi) :
  12 ≤ (9 * (x ^ 2 * Real.sin x ^ 2) + 4) / (x * Real.sin x) := by
  /-
  To find the minimum value of (9x^2 sin^2 x+4)/(x sin x) for 0 < x < π, we need to show that it is at
    least 12. We start by noting that the expression can be rewritten using the
    division property of inequalities. We then use the fact that sin x and x are
    positive in the given range to establish the necessary inequalities. Finally, we
    apply these results to conclude that the minimum value is indeed 12.
  -/
  -- We start by ensuring that the product x * sin x is positive in the given range.
  have h : 0 < x * Real.sin x := by
    apply mul_pos
    -- x is positive in the range (0, π).
    exact h.1
    -- sin x is positive in the range (0, π).
    exact Real.sin_pos_of_pos_of_lt_pi h.1 h.2
  -- Using the division property of inequalities, we rewrite the expression.
  rw [le_div_iff h]
  /- tactic state:
    x : ℝ
    h : 0 < x ∧ x < π
    h : 0 < x * x.sin
    ⊢ 12 * (x * x.sin) ≤ 9 * (x ^ 2 * x.sin ^ 2) + 4
  -/
```

输出:

```
  -- This is equivalent to showing that 9x^2 sin^2 x - 12x sin x + 4 ≥ 0, and the left-hand
    side can be rewritten as a perfect square (3x sin x - 2)^2.
  -- We use the fact that (3x sin x - 2)^2 is non-negative to establish this.
  nlinarith [sq_nonneg (3 * x * Real.sin x - 2)]
```

在此示例中，证明补全模型首先从战术状态中提取目标表达式，*i.e.*，注释块最后一行的不等式 $12(x\sin x) \leqslant 9(x^2\sin^2 x)+4$，随后将其整理为 $9x^2\sin^2 x - 12x\sin x + 4 \geqslant 0$ 的形式，作为思维链推理过程的一部分。最后，将左侧表达式重写为完全平方项以证明其非负性，从而完成证明。

B. MiniF2F-test 题目示例解答

为阐明非思维链（non-CoT）模式与思维链（CoT）模式提示的不同优势及其互补性，我们提供了非 CoT 模式成功而 CoT 模式失败的示例，反之亦然。

B.1. 非 CoT 模式优于 CoT 模式的问题

```
theorem induction_pord1p1on2powklt5on2 (n :  ) (h  : 0 < n) :
    (Π k in Finset.Icc 1 n, 1 + (1 :  ) / 2 ^ k) < 5 / 2 := by
  rw [Finset.Icc]
  simp_all [Nat.succ_le_iff, Nat.one_le_iff_ne_zero]
  have h  : 0 < 2 ^ k := by apply pow_pos <;> norm_num
  norm_num
  have h  : (2 :  )  0 := by norm_num
  field_simp
  rw [div_lt_div_iff]
  ring_nf
  norm_cast
  nlinarith
  all_goals norm_cast
  all_goals linarith
```

```
theorem imo_1960_p2 (x :  ) (h  : 0  1 + 2 * x) (h  : (1 - Real.sqrt (1 + 2 * x)) ^ 2
    0)
    (h  : 4 * x ^ 2 / (1 - Real.sqrt (1 + 2 * x)) ^ 2 < 2 * x + 9) : -(1 / 2)  x  x <
    45 / 8 := by
  norm_num at h  h  h
  have h  : 0  1 + 2 * x := by linarith
  have h  : 0 < 1 + Real.sqrt (1 + 2 * x) := by
    nlinarith [Real.sqrt_nonneg (1 + 2 * x)]
  have h  : 4 * x ^ 2 / (1 - Real.sqrt (1 + 2 * x)) ^ 2 < 2 * x + 9 := by linarith
  have h  : 1 - Real.sqrt (1 + 2 * x)  0 := by
    intro h
    apply h
    nlinarith
  have h  : 4 * x ^ 2 / (1 - Real.sqrt (1 + 2 * x)) ^ 2 = (1 + Real.sqrt (1 + 2 * x))
    ^ 2 := by
    field_simp [h ]
    nlinarith [sq_sqrt (show 0  1 + 2 * x by linarith)]
  rw [h ] at h
  constructor <;> nlinarith [sq_sqrt (show 0  1 + 2 * x by linarith)]
```

B.2. CoT 模式优于非 CoT 模式的问题

```
theorem mathd_algebra_459 (a b c d : ) (h : 3 * a = b + c + d) (h : 4 * b = a + c +
    d)
    (h : 2 * c = a + b + d) (h : 8 * a + 10 * b + 6 * c = 24) : ↑d.den + d.num = 28
    := by
  /-
  Given the system of equations involving the times that Anna (a), Bertram (b), Carli
    (c), and David (d) can hold their breaths:
  1. 3a = b + c + d
  2. 4b = a + c + d
  3. 2c = a + b + d
  4. 8a + 10b + 6c = 24
  We need to find the sum of the numerator and the denominator of the fraction
    representing David's time in minutes, simplified to its lowest terms, and show
    that it is 28.
  First, we solve the system of equations step by step:
  - From the first equation, we can express d in terms of a, b, and c:
    d = 3a - b - c
  - Substitute d into the second equation:
    4b = a + c + (3a - b - c)
    Simplifying, we get:
    4b = 4a - b
    5b = 4a
    b = 4a/5
  - Substitute b into the third equation:
    2c = a + 4a/5 + (3a - b - c)
    Simplifying, we get:
    2c = a + 4a/5 + 3a - 4a/5 - c
    2c = 4a - c
    3c = 4a
    c = 4a/3
  - Substitute b and c into the fourth equation:
    8a + 10(4a/5) + 6(4a/3) = 24
    Simplifying, we get:
    8a + 8a + 8a = 24
    24a = 24
    a = 1
  Now, we find b and c:
  - b = 4a/5 = 4/5
  - c = 4a/3 = 4/3
  Finally, we find d:
  - d = 3a - b - c = 3 · 1 - 4/5 - 4/3 = 3 - 4/5 - 4/3 = 45/15 - 12/15 - 20/15 = 13/15
```

```
  The fraction 13/15 is already in its simplest form, and the sum of the numerator and
    the denominator is:
  13 + 15 = 28
  -/
  have h  : a = 1 := by
    have h  : 8 * a + 10 * b + 6 * c = 24 := h
    have h  : 24 * a = 24 := by
      -- Solve the system of equations to find the value of a
      linarith
    exact mul_right_cancel  (show (24 :  )   0 by norm_num)
      (by linarith)
  have h  : b = 4 / 5 := by
    -- Solve for b using the value of a
    linarith
  have h  : c = 4 / 3 := by
    -- Solve for c using the value of a
    linarith
  have h  : d = 13 / 15 := by
    -- Solve for d using the values of a, b, and c
    linarith
  -- Sum the numerator and the denominator of d
  norm_num [h ]
  <;> rfl
```

```
theorem numbertheory_x5neqy2p4 (x y :  ) : x ^ 5   y ^ 2 + 4 := by
  /-
  To show that for any two integers x and y, x^5 ≠ y^2 + 4, we can use modular arithmetic
    to demonstrate that the equation x^5 ≡ y^2 + 4 (mod 11) has no solutions.
    Specifically, we will consider the possible values of x and y modulo 11 and show
    that none of these combinations satisfy the equation.
  -/
  -- Use modular arithmetic to show that the equation has no solutions modulo 11.
  have : ¬(x ^ 5 % 11 = (y ^ 2 + 4) % 11) := by
    -- Consider all possible values of x modulo 11.
    have : x % 11 = 0   x % 11 = 1   x % 11 = 2   x % 11 = 3   x % 11 = 4   x % 11 = 5
        x % 11 = 6   x % 11 = 7   x % 11 = 8   x % 11 = 9   x % 11 = 10 := by
      omega
    -- Consider all possible values of y modulo 11.
    have : y % 11 = 0   y % 11 = 1   y % 11 = 2   y % 11 = 3   y % 11 = 4   y % 11 = 5
        y % 11 = 6   y % 11 = 7   y % 11 = 8   y % 11 = 9   y % 11 = 10 := by
      omega
    -- For each combination of x and y modulo 11, check if the equation holds.
    rcases this with (h | h | h | h | h | h | h | h | h | h | h) <;>
```

```
    rcases this with (h' | h' | h' | h' | h' | h' | h' | h' | h' | h' | h') <;>
      simp [h, h', pow_succ, Int.mul_emod, Int.add_emod]
  -- Use the result from modular arithmetic to conclude the theorem.
  omega
```

```
theorem amc12_2000_p12 (a m c : ℤ) (h : a + m + c = 12) :
    a * m * c + a * m + m * c + a * c ≤ 112 := by
  /-
  Given nonnegative integers A, M, and C such that A + M + C = 12, we need to find the
    maximum value of A·M·C + A·M + M·C + A·C.
  To solve this, we can use the fact that the expression A·M·C + A·M + M·C + A·C is
    a polynomial in terms of A, M, and C. We can try different values of A, M, and C
    that sum to 12 and compute the value of the polynomial for each combination. The
    maximum value of the polynomial will then be the solution.
  -/
  -- We use a lemma that states if the sum of three numbers is less than or equal to
    a certain value, then the product of these numbers is also less than or equal to
    a certain value.
  have h : a ≤ 12 ∧ m ≤ 12 ∧ c ≤ 12 := by
    constructor
    · -- Since a + m + c = 12, a cannot be greater than 12.
      omega
    · constructor
      · -- Similarly, m cannot be greater than 12.
        omega
      · -- And c cannot be greater than 12.
        omega
  -- Using the lemma, we compute the value of the polynomial for different values of
    a, m, and c that sum to 12.
  cases' h with h _left h _right
  cases' h _right with h _middle h _right
  -- We use interval_cases to try different values of a, m, and c.
  interval_cases a <;> interval_cases m <;> interval_cases c <;>
    -- For each combination, we compute the value of the polynomial and check if it
    is less than or equal to 112.
    simp_all (config := {decide := true})
```