ROCQSMITH：自动优化能否打造更优秀的证明代理?

Andrei Kozyrev[1], **Nikita Khramov**[1],**Denis Lochmelis**[1],**Valerio Morelli**[1]**Gleb Solovev**[1], **Anton Podkopaev**[2,3][1]JetBrains Research Germany, [2]JetBrains Research Netherlands, [3]不来梅建造者大学

arXiv:2602.05762v1 [cs.AI] 5 Feb 2026

摘要

本研究探讨了自动 AI 智能体优化方法在形式化验证场景中的真实智能体上的适用性，并聚焦于 Rocq 领域中具有代表性且充满挑战的自动定理证明任务。我们评估了不同自动智能体优化器用于优化 Rocq 证明生成智能体时的表现，并考察智能体系统的精细调优环节（如提示设计、上下文知识和控制策略）是否能够自动化。结果表明，尽管有几种优化器带来了可测量的改进，但简单的少样本引导始终是效果最稳定的方法；然而，所研究的方法都无法达到经过精心设计的最先进证明智能体的性能。

1 引言

生成式人工智能（AI）的最新进展显著加快了软件开发流程。与此同时，实证研究表明，在真实的软件项目中，由大语言模型生成或辅助编写的代码仍可能存在安全弱点和正确性问题（Fu 等人，2025）。如果规约定义良好，形式化方法便能通过为生成的软件提供强有力的正确性保证和明确的验收标准，帮助降低这些风险。交互式定理证明器（ITP）支持形式化规约的开发和机器检查的证明。目前已有多种成熟的 ITP，包括 Rocq（前身为 Coq）（Bertot & Cast´eran，2013）、Lean（De Moura 等人，2015）、Agda（Kokke 等人，2020）和 Isabelle（Nipkow 等人，2002）。其中，Rocq 是一个尤为成熟的系统，已持续开发 30 多年，并在重要的验证工作中积累了卓著的记录，其中包括对 CompCert C 编译器的形式化验证（Leroy 等人，2016）。CompCert 是唯一一款得到广泛使用、且有一项全面研究报告称其不存在错误编译缺陷的 C 编译器（Yang 等人，2011）。

形式化软件验证仍是一项要求很高的工作，需要大量人类专业知识，因此，研究者为自动化 Rocq 中的定理证明投入了大量精力。一个很有前景的方向是开发用于交互式定理证明的自主智能体系统（Xu & Odersky, 2026）。智能体系统很适合这一场景，因为它们的交互循环与人类 Rocq 程序员的工作方式十分相似。智能体会观察当前环境状态，进行中间推理，并选择要执行的操作。在 Rocq 中，构造证明遵循类似的过程：用户（或智能体）反复检查当前的证明状态（由当前目标及其上下文组成），并应用策略，将证明状态转化为更简单的子目标，直到证明完成。经典的*ReAct*（推理与行动）范式（Yao et al., 2023）与这一工作流程十分契合：它逐步交替进行对证明状态的推理和策略选择。

已有多项工作尝试构建用于交互式定理证明的自主智能体系统（Yang 等人，2023；Teodorescu 等人，2024年；Kozyrev 等，2025）。其中，RocqStar（Kozyrev 等，2025）提出了一种用于生成 Rocq 证明的智能体系统，并在 IMM-300 基准测试上报告了强有力的实证结果。RocqStar 智能体由一条多组件流水线构成，该流水线结合了规划、执行与辅助控制机制，并依赖于大量精心设计的提示词和智能体交互。尽管此类系统展示了智能体方法在定理证明方面的潜力，但其设计与实现

则涉及相当大的工程投入。构建并调优一个生产级证明代理通常需要人工提示设计、任务分解、工具编排，以及对结构化上下文知识的整理。这促使我们研究自动智能体优化方法能否减少构建和适配此类智能体所需的人工工程量。

近期关于自动智能体优化的工作旨在通过将智能体置于反馈驱动的优化循环中，减少人工智能体工程。根据具体方法，优化的目标可以是提示词和少样本示例（Opsahl-Ong 等，2024）、外部上下文知识（Ouyang 等人，2025；Zhang 等人，2025），或智能体的控制流结构（Hu 等，2025）。现有对自动智能体优化方法的评估大多聚焦于基准驱动的任务，例如 SWE-Bench 或 AppWorld。然而，这些方法如何泛化到形式化领域中的智能体仍不明确；在这些领域中，正确性是被严格验证的，而非经验性测试。交互式定理证明是此类任务的一个代表性例子；它提供了一个强验证器，能够评估智能体的动作并从环境返回清晰的反馈信号，同时又构成一个无法以临时方式解决、需要结构化推理的挑战性问题。在本工作中，我们选取一个生产级 Rocq 证明智能体的简化但具代表性的核心，并将其作为测试用例来评估自动智能体优化方法。我们系统研究不同优化器在该设定下的表现，并评估它们将智能体提升至精心人工设计的基线性能水平的能力。

2 预备知识

RocqStar（Kozyrev 等，2025）是一个用于 Rocq证明生成的智能体系统，采用多组件流水线构建，将规划、执行、反思、评析以及基于检索的重新规划相结合，并依赖精心构造的提示词、辅助控制逻辑和丰富的工具接口，以便与证明器交互。尽管这一设计取得了出色的实证性能，但其构建、调优和维护需要大量人工工程工作。在本文中，我们省略这些手工设计的组件，并将一个简化的智能体核心作为基线。具体而言，我们研究一种单循环智能体拓扑：其中，一个最小化的控制结构反复观察当前证明状态，执行轻量级推理，选择工具调用，并执行该调用，直至终止。这种简化的智能体保留了大语言模型与 Rocq 环境之间的核心交互模式，同时避免引入额外的规划或控制模块。我们的目标是评估，自动智能体优化方法能否弥合这一最小化智能体与经过精心工程设计、由人工构建的智能体流水线之间的性能差距。

我们首先评估 DSPy 框架（Khattab 等，2023）中可用的自动智能体优化方法。我们在 DSPy 中实现了一个简化版的 RocqStar 智能体，并应用了一系列内置优化器，包括 *BootstrapFewShot*、*MIPROv2* （Opsahl-Ong 等，2024）、*SIMBA* 和 *GEPA*（Agrawal 等，2025）。*BootstrapFewShot* 通过收集训练集上的成功执行轨迹来构建少样本提示词，并将其中前 k 个示例（由于上下文限制，评估时使用 4 个）直接纳入提示词，为上下文学习提供了一个简单的基线。*MIPROv2* 通过提出多个指令候选项，并利用贝叶斯优化（Snoek 等，2012）根据任务表现对这些候选项的组合进行离散搜索，从而联合优化指令和少样本示例。*SIMBA* 对提示词变体进行随机搜索，保留能够提升任务表现的改动。*GEPA*采用进化优化策略来优化智能体的文本组件，例如提示词或指令，并通过选择机制和可选的反思反馈逐步提升表现。

除了以提示为中心的优化器，我们还考虑基于智能体过去的行为来优化其上下文知识的方法。*ACE* （Zhang 等人，2025）和 *ReasoningBank* （Ouyang 等人，2025）都通过在训练数据集上运行智能体，使用语言模型分析成功和失败的执行过程，并将由此得到的反思存储在一个可在测试时重复使用的结构化知识库中。这些知识涵盖了有关有效策略和常见失败模式的高层指导。两种方法的主要区别在于如何表示和检索这些上下文知识。*ReasoningBank* 维护一个过往经验记忆库，并在测试时使用编码器模型对当前查询进行嵌入，检索出最相似的记忆条目，再将其注入智能体的上下文中。*ACE* 则维护一个不断

Impl	优化时间	智能体	描述	模型	证明长度			总计
					≤4	5–8	9–20	
Koog	—	简单	—	GPT-4.1	10%	10%	0%	6%
Koog	45 分钟	简单	使用 BootstrapFewShot	GPT-4.1	24%	10%	6%	13%
Koog	3h	简单	搭配 ACE	GPT-4.1	22%	6%	2%	10%
Koog	2h	简单	搭配 RB	GPT-4.1	32%	14%	4%	17%
Koog	—	ReAct	—	GPT-4.1	34%	16%	10%	20%
Koog	45 分钟	ReAct	搭配 BootstrapFewShot	GPT-4.1	54%	28%	18%	**33%**
Koog	6h	ReAct	搭配 ACE	GPT-4.1	28%	16%	6%	17%
Koog	6h	ReAct	搭配 RB	GPT-4.1	40%	22%	12%	25%
Koog	6h	ReAct	使用 RB（无检索）	GPT-4.1	32%	20%	12%	21%
ADAS	70h	ADAS	—	GPT-4.1	30%	14%	8%	17%
DSPy	—	ReAct	—	GPT-4.1	38%	16%	4%	19%
DSPy	45 分钟	ReAct	搭配 BootstrapFewShot	GPT-4.1	66%	32%	22%	40%
DSPy	12h	ReAct	使用 MIPROv2	GPT-4.1	66%	42%	22%	**43%**
DSPy	14h	ReAct	使用 SIMBA	GPT-4.1	60%	42%	26%	**43%**
DSPy	32h	ReAct	使用 GEPA	GPT-4.1	30%	18%	4%	17%
Koog	—	SOTA	手写智能体	GPT-4.1	72%	56%	32%	**53%**

表 1：不同智能体优化方法在 Rocq 证明生成任务上的性能，按证明长度（策略数量）分组。*RB* 表示 *ReasoningBank*。

完善的智能体特定上下文，该上下文会在训练迭代中逐步优化，并在推理时完整提供给智能体，以便更好地利用可用的上下文窗口，而非依赖基于相似度的检索。最后，我们还评估了*ADAS*（Hu等人，2025），它优化的是智能体的控制流，而不是提示词或上下文。ADAS 将智能体表示为可执行代码，并通过向语言模型提供可用工具和任务描述，提示其生成定义智能体整体决策逻辑的代码，以此优化智能体的拓扑。

3 实验

我们研究了 Rocq 中面向智能体证明生成的自动优化方法，并回答两个研究问题。(i) 在以生产为导向的 Rocq 定理证明环境中，哪些自动 AI 智能体优化方法相较于固定基线智能体能够带来可测量的性能提升？(ii) 在 Rocq 环境中，经过自动优化的证明代理与最先进的、由人工设计的证明代理流水线相比表现如何?

我们使用先前关于证明代理的研究中引入的 IMM 定理证明基准，并复用 CoqPilot 发布的数据集划分（Kozyrev 等人，2024年）。该数据集包含来自 IMM 项目的 300 个定理（Podkopaev 等人，2019年）。我们依据人工编写的参考证明长度（以策略数量计）这一代理指标，按难度划分定理：较短的证明通常对应较简单的定理，而较长的证明则表明实例更困难。在评估中，我们从每个难度组中选取 50 个定理。我们另行分配 60 个定理用于训练优化器。由于实验设置存在差异，DSPy 实验所使用的确切训练子集与 Koog 实验所使用的子集略有不同（DSPy 的训练划分略偏向较简单的定理）。这不会影响每个框架内部的比较，因为其中的训练集是固定的，但在跨框架比较结果时可能会引入轻微差异。所有实验均使用 GPT-4.1，这是因为它具有较大的上下文窗口和可靠的工具使用能力。

所有实验均以第 2 节所述的简化版 RocqStar 智能体作为基线架构。我们使用两种实现对该智能体进行评估。在 *DSPy* 中，我们采用 *ReAct* 策略，该策略会在调用工具之前生成中间推理，并作为应用 DSPy 内置优化方法的基础。在 *Koog* 中，我们评估了 简单变体（该变体直接预测工具调用，而不生成显式推理轨迹）和*ReAct* 变体，以评估在此设置中显式推理是否是实现有效优化的必要条件。

我们对 § 2 中描述的优化方法进行训练，并在测试集上以固定资源预算进行评估，最多允许 20 次工具调用。我们报告各难度组（A–C）及总体（**总计**）的成功率（已解决定理的百分比），如表 1 所示。我们还报告优化时间，即优化器训练阶段所耗费的实际时间。

在表 1 中，*RB*表示*ReasoningBank*。除非另有说明，ReasoningBank 会在测试时根据编码器相似度检索排名前 5 的记忆条目，并将其注入智能体上下文。*RB*（不检索）会停用基于相似度的检索，转而将整个记忆库纳入上下文。对于*BootstrapFewShot*，由于上下文长度限制，我们最多使用四个示例。需要注意的是，对于 Koog 和 DSPy 中的 BootstrapFewShot，其绝对性能差异应谨慎解读，因为这些差异反映了框架特定差异，包括额外的隐式提示和工具处理逻辑。*GEPA* 采用重型优化模式运行，该模式支持结合反思反馈进行完整的进化搜索；而*MIPROv2*则采用轻型模式运行，因为重型配置的计算成本显著更高。[1] 对于*ADAS*，我们进行了十次优化运行，并报告其中表现最佳的智能体。

实验结果揭示了若干规律。少样本示例在所有评估设置中都能带来可靠的性能提升，因此是本研究中最稳健的优化方法。对于明确生成中间推理轨迹的智能体（如 ReAct 变体），少样本示例带来的提升尤为显著。同时，使用少样本示例时必须谨慎，因为较长的推理轨迹会迅速耗尽可用的上下文预算，并大幅增加令牌用量。对于以上下文为中心的方法，ReasoningBank 从记忆库中检索少量相关记忆条目，而非将整个记忆库注入提示中，这样更有利；与之相反，ACE 会将多达 200 条记忆注入上下文。这一观察与先前研究（Huang 等人，2025；Xu 等人，2024）的发现一致：未经筛选或相关性较弱的上下文可能引入噪声，从而降低模型性能。同时，有效使用 ReasoningBank 需要仔细考虑智能体输入：基于相似度的检索假设智能体输入在语义上具有利于编码的意义，而在我们的设置中，原始输入最初仅由定理标识符和文件路径构成，实际的证明状态则由环境稍后解析。因此，若要将上下文记忆方法应用于定理证明流水线，可能还需要额外的工程工作。

在以提示为中心的优化器中，我们发现，在这种设置下，更复杂的方法并不能始终优于更简单的基线方法。MIPROv2 可能适用于多组件流水线，在这类流水线中，对多个智能体节点上的指令与示例进行联合优化可能会带来益处；但对于本文研究的单提示智能体，其收益有限。SIMBA 和 MIPROv2 在 DSPy 中取得的提升与 BootstrapFewShot 相当，这表明观察到的大部分性能增益源于纳入了有效的少样本示例，而非更深入的提示演化。最后，ADAS 尝试通过生成可执行智能体代码来优化智能体的控制流，但其表现不稳定：尽管经过多轮优化迭代，我们并未观察到性能单调提升，且表现最佳的智能体很早便已出现。此外，生成的智能体对训练数据表现出严重依赖，并依靠脆弱的硬编码决策逻辑，因而泛化能力有限。

总体而言，所评估的优化器均未达到人工精心设计的最先进智能体的性能，这表明，针对形式化验证的智能体架构进行全自动优化仍是一项尚未解决的挑战。尽管如此，BootstrapFewShot 作为一种简单却出人意料地有效的优化方法脱颖而出，只需极少的工程投入便能持续提升性能。

4 结论

在本研究中，我们以简化的证明代理为评估环境，在 Rocq 定理证明任务上评估了多种自动智能体优化方法。结果表明，尽管若干优化器相较于固定基线取得了可衡量的提升，但简单的少样本自举是效果最稳定的方法，尤其适用于 ReAct 风格代理：在 Koog 中，整体成功率从 20% 提升至 33%；在 DSPy 中，则从 19% 提升至 40%。然而，所研究的方法均未达到经过精心设计的最先进证明智能体的性能。以上下文为中心的方法（如 ReasoningBank）在有选择地检索上下文信息时可能效果显著，但有时需要对流程进行细微适配。更复杂的优化器，包括 ADAS 等拓扑层面方法，在这一领域并未展现出稳定的收益。总体而言，我们的研究结果表明，当前的自动智能体优化技术

[1] MIPROv2 的重型模式留待未来实验。

可以在一定程度上减少手动调优的工作量，但尚不足以完全自动化设计用于形式化验证的高性能智能体系统。

参考文献

Lakshya A Agrawal、Shangyin Tan、Dilara Soylu、Noah Ziems、Rishi Khare、Krista Opsahl-Ong、Arnav Singhvi、Herumb Shandilya、Michael J Ryan、Meng Jiang、Christopher Potts、Koushik Sen、Alexandros G. Dimakis、Ion Stoica、Dan Klein、Matei Zaharia 和 Omar Khattab。GEPA：反思式提示演化能够胜过强化学习，2025。网址：https:// arxiv.org/abs/2507.19457。

Yves Bertot 和 Pierre Cast´eran。交互式定理证明与程序开发：*Coq'Art*：归纳构造演算。施普林格科技与商业媒体，2013。doi：10.1007/ 978-3-662-07964-5。

Leonardo De Moura、Soonho Kong、Jeremy Avigad、Floris Van Doorn 和 Jakob von Raumer。Lean 定理证明器（系统描述）。载于自动演绎——*CADE-25*：第 *25* 届国际自动演绎会议，德国柏林，*2015* 年 *8* 月 *1*—*7* 日，论文集 *25*，第 378–388 页。施普林格，2015。doi：https://doi.org/10.1007/978-3-319-21401-6 26。

Yujia Fu、Peng Liang、Amjed Tahir、Zengyang Li、Mojtaba Shahin、Jiaxin Yu 和 Jinfu Chen。《GitHub 项目中 Copilot 生成的代码的安全弱点：一项实证研究》。*ACM* 软件工程与方法学汇刊，34(8)，2025 年 10 月。ISSN 1049-331X。doi：10.1145/3716848。URL：https://doi.org/10.1145/3716848。

Shengran Hu、Cong Lu 和 Jeff Clune。《智能体系统的自动化设计》，2025。URL：https: // arxiv.org/abs/2408.08435。

Yue Huang、Yanbo Wang、Zixiang Xu、Chujie Gao、Siyuan Wu、Jiayi Ye、Xiuying Chen、Pin-Yu Chen 和 Xiangliang Zhang。《Breaking Focus：大语言模型中的上下文干扰诅咒》。*arXiv*，abs/2502.01609，2025。URL：https://api.semanticscholar.org/ CorpusID:276107466。

Omar Khattab、Arnav Singhvi、Paridhi Maheshwari、Zhiyuan Zhang、Keshav Santhanam、Sri Vardhamanan、Saiful Haq、Ashutosh Sharma、Thomas T. Joshi、Hanna Moazam、Heather Miller、Matei Zaharia 和 Christopher Potts。DSPy：将声明式语言模型调用编译为自我改进的流水线，2023年。URL https://arxiv.org/abs/2310.03714。

Wen Kokke、Jeremy G. Siek 和 Philip Wadler。Agda 编程语言基础。《计算机编程科学》，194:102440，2020年。ISSN 0167-6423。doi：https://doi. org/10.1016/j.scico.2020.102440。URLhttps://www.sciencedirect.com/science/ article/pii/S0167642320300502。

Andrei Kozyrev、Gleb Solovev、Nikita Khramov 和 Anton Podkopaev。CoqPilot：用于基于大语言模型生成证明的插件。载于 第*39*届 *IEEE/ACM* 自动化软件工程国际会议论文集，ASE '24，第2382–2385页，美国纽约州纽约市，2024年。计算机协会。ISBN 9798400712487。doi：10.1145/3691620.3695357。URL https://doi.org/10.1145/3691620.3695357。

Andrei Kozyrev、Nikita Khramov、Gleb Solovev 与 Anton Podkopaev。RocqStar：利用相似度驱动的检索与智能体系统进行 Rocq 生成，2025。URL https: //arxiv.org/abs/2505.22846。

Xavier Leroy、Sandrine Blazy、Daniel K¨astner、Bernhard Schommer、Markus Pister 与 Christian Ferdinand。CompCert——一部形式化验证的优化编译器。载于*ERTS 2016*：嵌入式实时软件与系统，第八届欧洲大会，2016。

Tobias Nipkow、Markus Wenzel 与 Lawrence C Paulson。*Isabelle/HOL*：面向高阶逻辑的证明助手。施普林格，2002。doi: https://doi.org/10.1007/3-540-45949-9 5。

Krista Opsahl-Ong、Michael J Ryan、Josh Purtell、David Broman、Christopher Potts、Matei Zaharia 和 Omar Khattab。《优化多阶段语言模型程序的指令与示例》，2024年。网址：https://arxiv.org/abs/2406.11695。

Siru Ouyang、Jun Yan、I-Hung Hsu、Yanfei Chen、Ke Jiang、Zifeng Wang、Rujun Han、Long T. Le、Samira Daruki、Xiangru Tang、Vishy Tirumalashetty、George Lee、Mahsan Rofouei、Hangfei Lin、Jiawei Han、Chen-Yu Lee 和 Tomas Pfister。《借助推理记忆扩展智能体自我演化》，2025。网址：https://arxiv.org/abs/2509.25140。

Anton Podkopaev、Ori Lahav 和 Viktor Vafeiadis。弥合编程语言与硬件弱内存模型之间的鸿沟。*ACM* 编程语言论文集，3(POPL):1–31，2019年。

Jasper Snoek、Hugo Larochelle 和 Ryan P. Adams。机器学习算法的实用贝叶斯优化，2012年。网址 https://arxiv.org/abs/1206.2944。Laetitia Teodorescu、Guillaume Baudart、Emilio Jes´us Gallego Arias 和 Marc Lelarge。NLIR：用于机械化定理证明的自然语言中间表示。发表于*MathAI@NeuRIPS 2024* - 第四届数学推理与人工智能研讨会，加拿大温哥华，2024年12月。网址 https://hal.science/hal-04886208。Xiaohan Xu、Chongyang Tao、Tao Shen、Can Xu、Hongbo Xu、Guodong Long、Jian guang Lou 和 Shuai Ma。重读有助于提升大语言模型的推理能力，2024年。网址 https: //arxiv.org/abs/2309.06275。Yichen Xu 和 Martin Odersky。智能体证明自动化：案例研究，2026年。网址 https: //arxiv.org/abs/2601.03768。Kaiyu Yang、Aidan Swope、Alex Gu、Rahul Chalamala、Peiyang Song、Shixing Yu、Saad Godil、Ryan J Prenger 和 Animashree Anandkumar。LeanDojo：基于检索增强语言模型的定理证明。神经信息处理系统进展，36:21573–21612，2023年。Xuejun Yang、Yang Chen、Eric Eide 和 John Regehr。发现并理解 C 编译器中的错误。发表于第 *32* 届 *ACM SIGPLAN* 编程语言设计与实现会议论文集，第 283–294 页，2011年。doi: https://doi.org/10.1145/1993498.1993532。Shunyu Yao、Jeffrey Zhao、Dian Yu、Nan Du、Izhak Shafran、Karthik Narasimhan 和 Yuan Cao。ReAct：在语言模型中协同推理与行动，2023年。网址 https://arxiv. org/abs/2210.03629。Qizheng Zhang、Changran Hu、Shubhangi Upasani、Boyuan Ma、Fenglu Hong、Vamsidhar Kamanuru、Jay Rainton、Chen Wu、Mengmeng Ji、Hanchen Li、Urmish Thakker、James Zou 和 Kunle Olukotun。智能体上下文工程：为自我改进语言模型演化上下文，2025年。网址 https://arxiv.org/abs/2510.04618。