计算机科学
COMPUTER SCIENCE

基于预训练语言模型和合一算法的自动定理证明

陈红休, 曾霞, 刘志明, 赵恒军

引用本文

陈红休, 曾霞, 刘志明, 赵恒军. 基于预训练语言模型和合一算法的自动定理证明[J]. 计算机科学, 2026, 53(4): 40-47.

CHEN Hongxiu, ZENG Xia, LIU Zhiming, ZHAO Hengjun. Automatic Theorem Proving Based on Pre-trained Language Models and Unification [J]. Computer Science, 2026, 53(4): 40-47.

相似文章推荐（请使用火狐或 IE 浏览器查看文章）

Similar articles recommended (Please use Firefox or IE to view the article)

基于循环一致性约束的LLM增强型语言模型训练框架

LLM-augmented Training Framework with Cycle-Consistency Constraints

计算机科学, 2026, 53(4): 377-383. https://doi.org/10.11896/jsjkx.250600032

Agent4Stu:基于大语言模型的学生作答行为高效模拟智能体

Agent4Stu:Efficient LLM-based Student Answer Behavior Simulation Agent

计算机科学, 2026, 53(4): 347-355. https://doi.org/10.11896/jsjkx.250800012

基于点态流形与一致正则的半监督学习算法

Semi-supervised Learning Algorithm Based on Pointwise Manifold Structures and Uniform Regularity Constraints

计算机科学, 2026, 53(4): 173-179. https://doi.org/10.11896/jsjkx.250300086

大模型驱动的形式化定理证明:综述与展望

Formal Theorem Proving Empowered by Large Language Model:Survey and Perspectives

计算机科学, 2026, 53(4): 1-23. https://doi.org/10.11896/jsjkx.251000067

基于大型语言模型增广的少样本持续毒性检测

Few-shot Continuous Toxicity Detection Based on Large Language Model Augmentation

计算机科学, 2026, 53(3): 321-330. https://doi.org/10.11896/jsjkx.250600010

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.251000066

基于预训练语言模型和合一算法的自动定理证明

陈红休　曾　霞　刘志明　赵恒军
西南大学计算机与信息科学学院 软件学院　重庆 400715
(chx666@email.swu.edu.cn)

摘　要　预训练语言模型在 Metamath 等形式化环境中,在证明定理方面展现出显著潜力。然而,这种潜力尚难以稳定转化为可靠的推理能力。现有方法通常要求语言模型直接预测替换项,而这些替换项来自一个开放且潜在无限的表达式空间,缺乏逻辑约束,导致模型泛化能力受限。为此,提出一种新的证明范式:在推理过程中引入工作变量以临时替代具体项,并通过合一算法推导出其具体实例。该设计使得模型能够聚焦于定理选择,而无需生成缺乏逻辑指导的替换。基于这一范式,从 Metamath 数学库中提取数据并构建 UnProver(Unification-Driven Prover),并在此数据上对其进行训练。此外,设计多种数据增强策略,进一步提升 UnProver 的性能。实验结果表明,UnProver 在测试集上整体优于包括 GPT-4o 在内的基线方法,在证明能力与效率方面均取得更优表现。此外,UnProver 还发现了 6 个新的、更简洁的证明,这些证明已被正式收录至 Metamath 官方数学库。

关键词:语言模型;定理证明;Metamath;合一;工作变量
中图分类号　TP181

Automatic Theorem Proving Based on Pre-trained Language Models and Unification

CHEN Hongxiu,ZENG Xia,LIU Zhiming and ZHAO Hengjun
School of Computer and Information Science,School of Software,Southwest University,Chongqing 400715,China

Abstract　Pre-trained language models(PLMs) have demonstrated considerable potential in proving theorems within formal environments such as Metamath. However,this potential remains difficult to translate into reliable reasoning ability. Existing approaches typically require PLMs to directly predict substitutions,which come from an open and potentially infinite expression space. The absence of logical constraints in this process limits the generalization capability of the models. To address this challenge,a new proof paradigm is introduced:work variables temporarily substitute concrete terms during inference,and their specific instances are derived through the unification algorithm. This design enables the model to focus on theorem selection rather than generating unguided substitutions. Based on this paradigm,a dataset is extracted from the Metamath library to construct UnProver(Unification-Driven Prover),which is trained on this data. In addition,several data augmentation strategies are designed to further enhance UnProver's performance. Experimental results show that UnProver consistently outperforms baseline methods and GPT-4o on test sets,achieving superior performance in both proving capability and efficiency. Moreover,UnProver discoveries six new and shorter proofs,which have been formally accepted into the official Metamath library.

Keywords　Language models,Theorem proving,Metamath,Unification,Work variables

1　引言

自动定理证明(Automatic Theorem Proving,ATP)是形式化方法研究的核心方向之一,其目标是在给定的时间约束内,自动生成以形式逻辑表达的定理的证明。所生成证明的正确性需在形式化环境中得到验证,这类环境包括 Lean[1],Coq[2]和 Isabelle[3]等。

近年来,以预训练语言模型(Pre-trained Language Models,PLMs)为代表的人工智能技术展现出强大的语义理解与生成能力,为 ATP 提供了全新的研究范式[4]。该范式允许语言模型通过学习海量数学文本中的推理模式,根据当前的证明状态预测证明策略(推理步骤),并在形式化环境中验证其正确性,再结合搜索算法完成定理的证明。

尽管近年来大多数自动定理证明依托于 Lean 等主流形式化系统,但不同的形式化系统在逻辑表达层级与推理机制上存在显著差异。特别地,Metamath[5]以其极简的推理规

到稿日期:2025-10-16　返修日期:2026-01-18
基金项目:国家自然科学基金(62272397,62472362,92582113);重庆市自然科学基金(CSTB2025NSCQ-LZX0019)
This work was supported by the National Natural Science Foundation of China(62272397,62472362,92582113) and Natural Science Foundation of Chongqing,China(CSTB2025NSCQ-LZX0019).
通信作者:赵恒军(zhaohj2016@swu.edu.cn)

则，为研究语言模型的符号推理能力提供了独特视角。与 Lean 丰富的 tactic 机制不同，Metamath 拥有更底层的结构，使得每一步推理均可被独立验证。因此，研究 Metamath 不仅有助于探索语言模型在自动证明中的泛化能力，还能够作为 Lean 等系统的有力补充。

现有的基于 Metamath 的策略预测研究[4,6-7]多采用端到端的方法：在给定证明目标时，语言模型直接预测包含定理与替换的策略。然而，替换项往往无法直接从上下文中推断，并且其属于一个开放且潜在无限的表达式空间。这种策略缺乏显式的符号推理，导致模型泛化性能受限。

针对上述问题，本文提出 UnProver（Unification-Driven Prover）（见图 1 右下）。具体而言，在推理过程中引入工作变量，以占位暂未实例化的项；预测定理后，合一算法[8]将这些工作变量解析为具体实例，从而生成一个完全形式化的证明。这种分工机制使得语言模型能够专注于语义理解与证明结构的预测，而将替换项的推导交由合一算法完成。

为确保后续方法在形式化语义上具有严格的可验证性，本文在 Metamath 的希尔伯特风格逻辑体系上展开研究。在实验部分，基于 Metamath 数学库构建专用基准数据集，并设计多种数据增强策略以缓解形式化证明数据的稀缺问题（见图 1 右上）。实验结果表明，UnProver 在 Metamath 测试集与合成测试集上均显著优于基线方法，证明成功率与效率均有提升；同时，生成的证明更简洁且具有创新性。消融实验结果进一步表明，数据增强策略显著提升了模型泛化能力，本文提出的证明范式在推理效率与证明能力方面亦表现出明显优势。

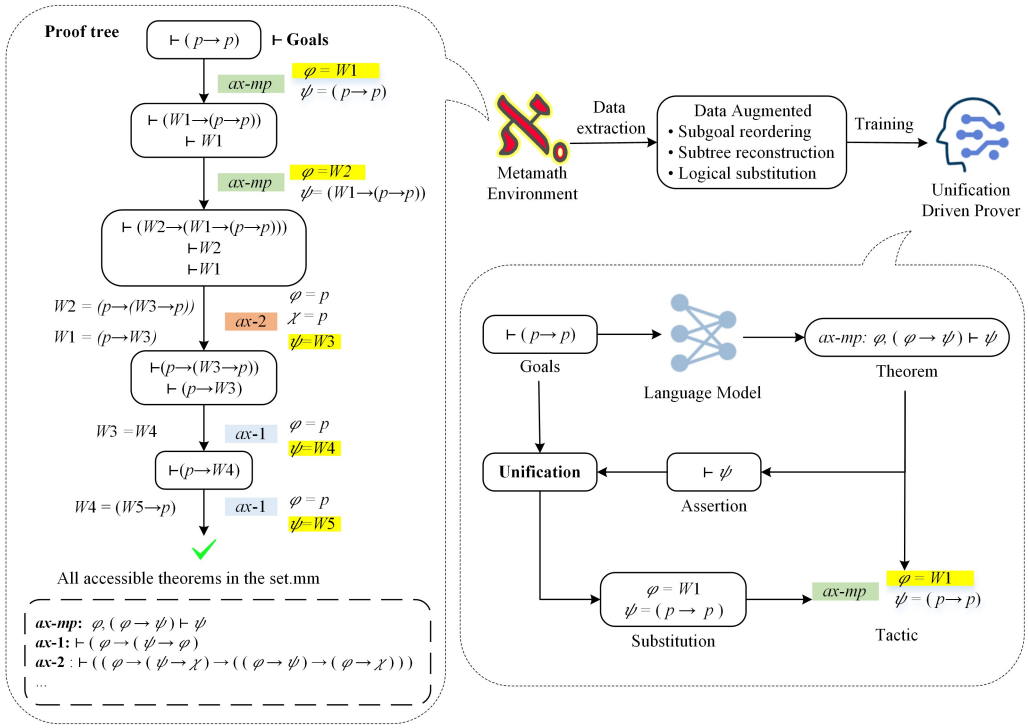

图 1 UnProver 整体工作流程

Fig. 1 Overall workflow of UnProver

本文的主要贡献如下：

1）提出一种新的证明范式：引入工作变量占位并结合合一算法，使模型专注于定理选择而非替换项生成。

2）数据集构建与增强：基于 Metamath 构建高质量形式化证明数据集，并设计系统化数据增强策略以缓解数据稀缺。

3）实验验证与应用价值：通过消融与对比实验验证了所提方法的有效性，并且 UnProver 发现了更短的证明（如 3jaob[1)]，bian1d[2)] 等），这些证明已被收录至 Metamath 官方库。

2 相关工作

2.1 语言模型驱动的定理证明

近年来，语言模型[9]在数学推理领域取得显著进展，如解决数学文字题求解[10]、数学问答[11]以及参数化偏微分方程建模[12]等任务。在诸多数学推理任务中，ATP 仍是最具挑战性的研究方向之一。

最新的语言模型驱动 ATP 的方法主要遵循 GPT-f[4]范式：在给定当前证明目标的情况下，利用语言模型预测可行的证明策略，并结合启发式的最佳优先树搜索来完成定理证明。后续研究在多个方向展开探索，包括数据增强、基于内核级证明项的推理流程优化[13]、新颖的证明搜索策略[14]（如蒙特卡罗树搜索[6]）、课程学习驱动的持续训练[15]以及检索增强方法[16]。随着 PLMs 能力的不断增强，其已能够仅依赖提示生成有效的证明策略。这些研究表明，语言模型在自动定理证明领域展现出巨大潜力。

然而，这些研究主要集中于模型训练范式与证明搜索策略的优化。例如，BFS-Prover[17]通过更强的示例联结能力与

1) https://us.metamath.org/mpeuni/3jaob.html

2) https://us.metamath.org/mpeuni/bian1d.html

归一化的最优优先搜索来提升推理效率；DeepSeek-Prover[18]依托于更大规模的数据和思维链机制，增强定理选择能力。但这些方法依旧停留在模型层与搜索层的改进，未触及证明的底层逻辑结构本身。相比之下，本文方法改变了证明的构造方式，将符号推理直接嵌入推理流程，使证明过程在结构化程度与泛化能力上实现显著提升。

2.2 Metamath 中的策略生成

在 ATP 中，策略生成是核心挑战之一[19]，其目标是预测一个或多个证明策略，以逐步构建完整的定理证明。在 Metamath 系统中，每个策略由一个定理与一个替换组成。已有系统如 Holophrasm[20]与 MetaGen[7]，将证明目标编码为语法树结构输入神经网络，以预测证明策略；而后续工作[4,21]则将证明目标表示为文本，并利用语言模型进行策略生成。

然而，这些方法均要求模型同时预测定理及其替换。本文方法仅需模型预测合适的定理，并通过合一算法在逻辑推理过程中自动求解相应的替换，从而避免了模型直接面对开放式替换空间的困难。此外，将带有工作变量的证明状态直接输入模型，使其能够在更底层的推理结构中进行学习与推断，从而在遇到新定理时更容易迁移出有效的证明思路。

3 背景知识

3.1 希尔伯特系统$\mathcal{H}$

希尔伯特系统[22]，又称希尔伯特风格演绎系统，是经典逻辑的一种基础性证明体系。其特点在于仅依赖少量的公理模式与推理规则，被认为是最早且最具影响力的逻辑形式化方法之一。希尔伯特公理系统$\mathcal{H}$由形式语言和演绎系统组成，其中$\mathcal{H}$的形式语言包括一个可数无限的字母表和合式公式。$\mathcal{H}$的演绎系统通常包括：

1)公理模式(Axiom Scheme)：

$ax\text{-}1$：$\vdash(\varphi\rightarrow(\psi\rightarrow\varphi))$

$ax\text{-}2$：$\vdash((\varphi\rightarrow(\psi\rightarrow\chi))\rightarrow((\varphi\rightarrow\psi)\rightarrow(\varphi\rightarrow\chi)))$

$ax\text{-}3$：$\vdash((\neg\varphi\rightarrow\neg\psi)\rightarrow(\psi\rightarrow\varphi))$

2)推理规则：$ax\text{-}mp$(Modus Ponens)：

$$\frac{\vdash\varphi \quad \vdash(\varphi\rightarrow\psi)}{\vdash\psi}$$

若已知公式φ和$\varphi\rightarrow\psi$，则可以推出公式ψ。

定义 1(证明) 设Γ为一个$\mathcal{H}$公式集合，称合式公式序列$\varphi_1,\cdots,\varphi_n$是一个基于$\Gamma$的证明，如果对任意的$i(1\leqslant i\leqslant n)$，满足以下条件：

(1)φ_i是$\mathcal{H}$的公理；

(2)$\varphi_i\in\Gamma$；

(3)φ_i是通过对该公式序列中排在φ_i前面的两个合式公式使用$ax\text{-}mp$规则而得到的。

在这种情况下，称$\Gamma\vdash\varphi_n$为一个定理(推理规则)，这段合式公式被称为$\Gamma\vdash\varphi_n$的证明，Γ中的公式被称为$\Gamma\vdash\varphi_n$的假设，而φ_n被称为$\Gamma\vdash\varphi_n$的结论。

公理模式与推理规则各自对应无穷多条具体实例，其内部的变量称为元变量，用以表示任意合式公式，可通过替换生成具体实例。

定义 2(替换) 项(公式)对变量的一个替换是一个集合$\{p_1=t_1,\cdots,p_n=t_n\}$，其中每个$p_i$是不同的命题变量，每个$t_i$是一个项，且$t_i$不与对应的变量$p_i$相同。空的替换是空集。小写希腊字母$\{\lambda,\mu,\delta,\theta\}$将用来表示替换。

定义 3(策略) 一个策略是一个二元组(t,θ)，其中t是一个定理$\Gamma\vdash\varphi$，θ是一个替换，并且满足$\theta(\varphi)=G$，即对t的结论应用θ后匹配某个目标G。

定义 4(目标) 设$\varphi_1,\cdots,\varphi_n$是某个定理的一个证明序列。如果对某个$i(1\leqslant i\leqslant n)$，$\varphi_i$在其证明的上下文中是待证命题$G$，则称$G$的该步的目标。

定义 5(子目标) 给定一个应用于目标G的策略$(\Gamma\vdash\varphi,\theta)$，其子目标为$\{\theta(h)\mid h\in\Gamma\}$，即该定理的假设在替换$\theta$下的结果。

3.2 交互式定理证明器 Metamath

Hilbert 风格系统提供了一种通用的形式化演绎体系，用于定义哪些证明是有效的；而 Metamath 则给出该体系的一个极简且高度可验证的计算机实现。

Metamath 是目前应用广泛的形式化证明系统之一，主要用于证明验证，并且能够同时与人类和语言模型进行交互。其核心目标是通过简单的替换操作来刻画严格的数学推理。Metamath 的数学库被称为 set.mm，汇集了 46 521 条由人类撰写的、基于 ZFC 集合论的定理及证明。

3.2.1 Metamath 中的证明

在 Metamath 中，采用自底向上的证明方式，从一个待证明的目标出发，不断应用策略，将其逐步归约为公理或已被证明且假设可验证为真的定理。策略中涉及的定理均来自 set.mm。

如图 2 所示，为构建命题$\vdash(p\rightarrow p)$的证明树，第一步策略使用$ax\text{-}mp$及其替换，生成两个待证明的子目标。由于一个定理可能包含多个假设，因此证明过程自然形成树结构。

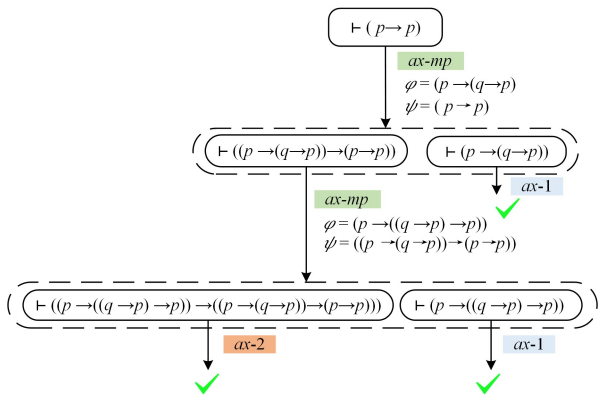

图 2 Metamath 中定理 id 的证明树

Fig. 2 Proof-tree for theorem id in Metamath

3.2.2 替换

给定一个形如$\Gamma\vdash\varphi$的定理t，其中的变量可分为以下两类。

1)受约束变量：结论φ中的变量。

2)非约束变量：出现在Γ中某个假设但不在结论φ中的变量。

相应地，这两类变量的替换分别称为受约束替换与非约束替换。需要注意，非约束替换无法从上下文中推断，而受约

束替换则可以通过将结论 φ 与当前证明目标进行合一而唯一确定。

例如，在图 2 的第一个策略中，$\varphi=(p\rightarrow(q\rightarrow p))$是一个非约束替换，无法从上下文推断；而受约束替换 $\psi=(p\rightarrow p)$则是通过 $\vdash\psi$ 与 $\vdash(p\rightarrow p)$进行合一得到的。

3.3 合一算法

合一是形式逻辑中的一个经典算法，用于判断两个表达式是否可以通过变量替换而变得相同。它在众多自动定理证明系统中被广泛应用。

定义 6(合一子) 设 $U=\varphi_1,\cdots,\varphi_n$ 为一组表达式，若存在替换 θ，使得：

$$\theta(\varphi_1)=\theta(\varphi_2)=\cdots=\theta(\varphi_n)$$

则称 θ 是 U 的合一子。对于某个替换 λ，若合一子 μ 满足：任意 U 的合一子 θ 均可表示为

$$\theta=\lambda(\mu)$$

则称 μ 为 U 的最一般合一子(Most General Unifier，MGU)。

本文采用 Robinson 合一算法[8]。该算法针对两个表达式，若存在可使其相同的替换，则能够求解出该替换。其基本原理是：通过逐步将复合表达式分解为更简单的子方程，并将变量绑定到项，最终得到的结果即为 MGU，它能够涵盖所有其他可能的合一子。

4 方法

在本文方法中，策略预测仅要求 PLMs 为每个策略预测相应的定理。替换由合一算法完成：通过将定理的断言与当前证明目标进行合一，得到所需的替换；若存在尚无法立即推断出的项，则使用工作变量作为占位符进行临时替代。

4.1 动机

在定理证明过程中，选择合适的替换对于证明的效率与成功率至关重要。若选择不当的替换，可能导致部分子目标在当前证明目标下易于证明，而其他子目标的证明则变得更难，甚至无法证明。图 3 展示了两种不当的非约束替换，这些替换会导致证明失败，甚至引发循环推理。

然而，这类非约束的替换往往无法从上下文中推断，因为它们属于一个开放且潜在无限的表达式空间。因此，如何生成合适的、能够推动证明进展的实例，成为一项具有挑战性的任务。

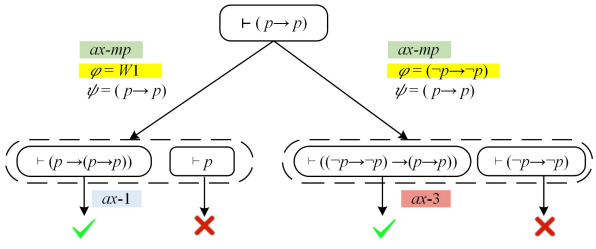

图 3 不恰当的非约束替换

Fig. 3 Inappropriate unconstrained substitution

4.2 工作变量

为应对这一挑战，本文在合一过程中引入工作变量作为无法立即推断出的替换项的占位符。通过这种方式，可以将替换的选择延后，直到获得足够的结构信息，再利用合一算法在后续证明步骤中实例化这些工作变量。

工作变量作为元变量，以符号“W”开头并跟随一个自然数来表示。它们可由命题逻辑中的任意合式公式实例化。此外，不同的工作变量可被实例化为相同的公式。

在合一过程中，当定理中的某些变量无法立即推断时，将其替换为工作变量，通常包括以下两类情况：

1)所有非约束变量；

2)由于当前目标中存在工作变量而无法实例化的约束变量。

当一个包含非约束变量的定理被作为策略应用于当前证明目标时，生成的子目标会自然携带工作变量。在下一步策略应用中，这些包含工作变量的子目标将与定理断言进行合一，得到 MGU，其中既包含受约束变量的替换(记作 θ')，也包含工作变量的替换(记作 λ^W)。随后，将替换 λ^W 应用于尚未完成的子目标，从而将先前引入的工作变量实例化为具体公式。

如图 4 所示，解析出工作变量的替换 $W1=(p\rightarrow W3)$，$W2=(p\rightarrow(W3\rightarrow p))$，并将其应用于未完成的子目标 $W1$ 与 $W2$(虚线灰色箭头所示)。由于无法推断 ψ，进一步引入一个新的工作变量 $W3$。

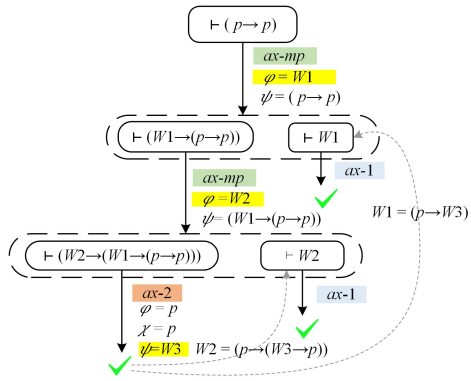

图 4 工作变量代替的证明树

Fig. 4 Proof tree substituted by working variables

需要注意的是，在所有子目标均被证明完成后，可能仍然存在未被实例化的工作变量。此时，这些工作变量可以被替换为任意公式，而不会影响整个证明的正确性。

4.3 策略预测

本文采用 PLMs 来预测策略中所需的定理。然而，由于工作变量是在全局范围内进行替换，且只有所有子目标均被证明后才能判定一个目标被证明，因此必须考虑所有子目标的可证明性。

基于上述过程，将证明过程中的子目标集合 G 合并为证明树中的一个节点，并在每一步将策略应用于子目标序列中的第一个目标 $G[1]$。这样就能够重构证明树(见图 4)，并将其扩展为带有工作变量的证明树(见图 1 左边部分)。

模型的输入为一个包含工作变量的证明树节点(一组子目标序列)，输出则是与对应边相关联的定理。在每次预测时，模型会针对序列中的第一个子目标采样 $e=16$ 个候选定理，并为每个候选定理分配启发式分数 h。所有候选定理均符合 Metamath 的语法约束，对应于 set.mm 中已被形式化证明的定理，并过滤不可合一的定理。如图 5 所示，当输入

目标为 $\vdash(p\rightarrow p)$时，模型会预测出若干候选定理，如 *mpd* 与 *ax-mp*。在预测阶段，包含工作变量的目标被作为输入送入模型，其中工作变量能够引导语言模型生成更为合适的定理。

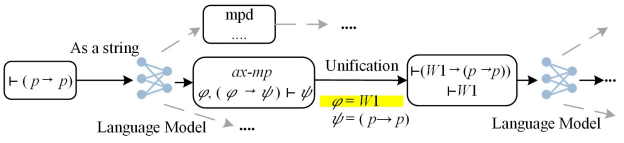

图 5 语言模型的策略生成

Fig. 5 Tactic prediction by PLMs

本文采用以定理为核心的训练目标，将其形式化为一个条件语言建模任务：给定一系列目标，预测应用的定理。准备训练数据时，将每个样例格式化为如下形式：

GOAL $(GOAL) STATEMENT $(STATEMENT)其中，$(·) 表示子目标序列的占位符；$(GOAL)表示当前待证明的子目标序列；$(STATEMENT)表示模型生成定理的形式化表达，即该子目标应当应用的具体定理；其他部分作为提示符。例如：

GOAL|-(ph->ph)STATEMENT ph,(ph->ps)|-ps

是一条完整的训练数据。在训练和推理阶段，GOAL $(GOAL)STATEMENT 为语言模型的输入，$(STATEMENT)作为语言模型的输出。例如，图 5 中第一次策略预测，模型的输入为：

GOAL |-(ph-> ph) STATEMENT

模型输出的候选定理为 ph,(ph-> ps)|- ps，对应推理规则 *ax-mp*。在推理过程中，可以多次采样生成多个候选定理，用于后续证明搜索。

4.4 构建数据集

本文基于 Metamath 形式语言及其数学库 set.mm 构建实验基准，专注于命题逻辑相关的子集，其中包含约 2083 条定理与 29 条公理。其中，训练集/验证集/测试集分别包含 1883/100/100 条定理。通过定理之间的引用关系构建有向无环图，并依据该结构划分数据，确保验证集或测试集中的定理不会用于证明其他定理。

对于每条定理，提取其原始证明(见图 2)，并构建带有工作变量的证明树；随后从该证明树中抽取(目标，策略)对，作为原始数据集中的基本训练样例。

测试集中仅包含 100 条命题逻辑定理，难以全面评估模型性能，因此需要额外构造模型从未见过的命题，构造合成数据集(Synthetic Test Set)。为此，本文利用自然数与命题逻辑定理之间的双射关系[23]，通过在自然数范围内均匀采样来生成新的命题。为保证训练集中的证明长度短于合成测试集中的证明，每个命题定理的运算符数量 n 被限定在 3 到 5 之间，并随机采样 2700 条重言式作为测试定理。最终得到的合成测试集包含在训练过程中从未出现过的结构形式，从而能够对模型的泛化能力进行严格评估。

为了评估模型在不同结构复杂度下的泛化能力，本文分别统计了合成测试集与 Metamath 训练集和测试集在表达式结构上的关键特征，包括变量数量、运算符数量、表达式树深度以及总符号数量，如表 1 所列。表 1 中的结果表明合成数据集的结构明显更复杂，更强调多变量与复杂推理模式。

表 1 合成测试集与 Metamath 数据集在表达式结构上的统计对比

Table 1 Structural statistics comparison between the synthetic test set and Metamath dataset

Dataset	Size	Avg. # Variables	Avg. # Operators	Avg. # Tree Depth	Avg. # Symbols
Synthetic Test Set	2689	**4.95**	**4.11**	**4.55**	**9.06**
Metamath Training Set	1983	3.93	3.52	3.76	7.46
Metamath Test Set	100	4.15	3.71	4.04	7.86

4.5 数据增强

为了进一步提高模型的泛化能力，本文基于提出的证明范式，提出了多种数据增强策略。

1)子目标重排序。由于在证明过程中总是优先将策略应用于第一个子目标，因此不同的子目标排列顺序将导致不同的策略选择。基于该特性，通过改变子目标的处理顺序来生成异质证明树(见图 6)。该方法显式建模不同推理路径之间的逻辑等价性，不仅丰富了训练数据的逻辑多样性，还增强了模型对证明结构变体的泛化能力。

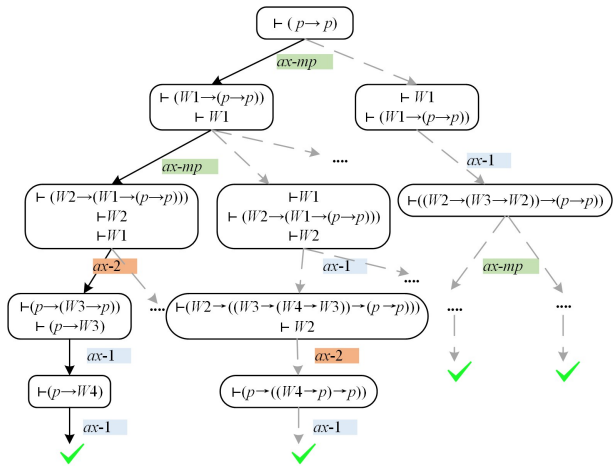

图 6 定理 id 的异构证明树

Fig. 6 Heterogeneous proof trees of theorem id

2)子树重构。Metamath 证明树中的每一棵子树都蕴含着一个完整的证明。针对每一个定理，从其证明树中选取中间节点作为新的根节点，并在此基础上重构带有工作变量的证明树，以形成增强后的训练样本。

需要注意的是，本文的两种增强策略均基于原始 Metamath 证明的语义等价变换。所有增强后的样本仍然经过 Metamath 的形式化校验，因此不会破坏原始证明结构的语义完整性。

4.6 UnProver 的证明搜索与算法描述

本文提出 UnProver(见算法 1)，该方法结合合一算法与 PLMs，用于证明给定的目标。在证明过程中，UnProver 在每一步会生成多个候选策略，并在限定的时间预算内，利用长度归一化的最佳优先搜索算法[17]来探索证明路径：若找到完整证明，则返回；否则，在超时或无可用策略时报告失败。

算法 1 UnProver：Unification-Driven Theorem Prover

输入：G_0 / * 初始证明目标 * /

输出：G_0 的证明或 Fail

1. 初始化优先级队列 Q←{({G_0},∅,0)} /*{证明目标，证明路径，优先级} */
2. 初始化工作变量的替换：μ^W
3. while time limit not exceeded ∧ Q do
4. (G,π,s)←pop(Q)
5. if G=∅ then return ProofFound (π,μ^W)
6. $[(\Gamma_1 \vdash \varphi_1, h_1),\cdots,(\Gamma_e \vdash \varphi_e, h_e)]$←ModelPredict(G,e)
7. for all i=1,…,e do
8. θ^c,λ^W←Unify(φ_i,G[1])
9. G'←RunTactic($\Gamma_i \vdash \varphi_i,\theta^c,\lambda^W$,G)
10. π_i←$\pi \cup \{(G[1],\Gamma_i \vdash \varphi_i)\}$
11. $s_i \leftarrow \frac{s+h_i}{|\pi|^{\alpha}}$ /*长度归一化*/
12. push(Q,(G',π_i,s_i))
13. add(μ^W,λ^W)
14. end for
15. end while
16. return Fail

长度归一化机制将归一化因子 α 纳入累积概率计算中，从而抑制搜索过程中对过深证明路径的偏向，有效平衡了探索与利用。

在证明搜索阶段，子目标的顺序由其熵值 $Ent(g)$ 决定：

$$Ent(g)=1-\frac{W(g)}{V(g)}$$

具体地，设 g 为一个目标，$V(g)$ 表示 g 中变量的总数，$W(g)$ 表示其中工作变量的数量。

5 实验与结果分析

5.1 实验设置

本实验使用的 PLMs 初始化自 Hugging Face 上的 google/byt5-small 检查点（2.99×10^8 参数量）。该模型是一个基于 T5[24] 的编码器-解码器式 Transformer[25]，直接在 UTF-8 字节序列上运行，无需额外分词步骤。字节级输入在逻辑推理场景中具有显著优势：逻辑表达式中包含大量符号元素，传统分词器可能将其误切分，从而影响模型的学习效果。而字节级建模可以完整保留其符号结构，避免额外的词表设计与切分误差，使模型能够更好地捕捉逻辑公式的精确语法模式。

在原始数据集上对模型进行监督微调，并在验证集损失达到最小值的轮次时，采用早停策略。模型采用 AdamW 优化器进行训练，实际有效 batch size 为 8。在前 2000 步中，学习率从 0 线性升至最大学习率，之后按照余弦退火（Cosine Decay）方式衰减至 0。最大学习率设定为 5×10^8，权重衰减系数为 0.01，并在训练中使用 1.0 的梯度裁剪阈值。

所有实验均在单张 NVIDIA RTX 3090 GPU（24 GB 显存）上完成，耗时约 1 天；评估过程则耗时 3 天。

5.2 对比实验

本实验在 Metamath 测试集与合成测试集上对 UnProver 的性能进行评估。评价指标采用 Pass@1：仅获得一次尝试机会，且必须在 5 分钟的时间限制内找到完整证明。

本文采用 Holophrasm 和 MetaGen 作为基线方法与 UnProver 进行对比。需要注意的是，Holophrasm 与 MetaGen 的训练覆盖 set.mm 中的全部定理，而 UnProver 则仅在命题逻辑子集进行训练。

另一个基线方法是将 GPT-4o[26] 用作策略生成器。在该方法中，向 GPT-4o 提供一个特定设计的提示模板。对于给定的证明目标，在零样本设定下调用 GPT-4o 生成 $e=16$ 个候选策略（$temperature=0$），并为每个定理分配一个 0 到 1 的置信度分数。最后结合长度归一化的最佳优先搜索整合有效步骤，以寻找完整证明。

遗憾的是，本文无法与目前的 SOTA，即 HTPS[6] 进行直接比较，因为其模型并未开源。为公平起见，在相同条件下复现其范式，并据此进行消融实验。

由表 2 可知，UnProver 在两个测试集上的表现均显著优于基线方法。这一结果表明，UnProver 不仅能够高效地解决来自训练分布的定理，还能够在面对全新结构时保持较强的性能。相比之下，即便考虑到 GPT-4o 可能因数据污染而接触过部分真实证明，其性能依然明显较弱。造成这一差异的主要原因在于证明范式的不同，基线方法要求模型直接在开放表达式空间中生成替换项。由于缺乏逻辑约束，模型易生成不合法或低相关性的候选，且难以兼顾上下文中所有子目标的可证明性，从而在复杂或全新结构下表现不稳定。

表 2 Metamath 测试集和合成测试集上的 Pass@1

Table 2 Pass@1 on the Metamath test set and synthetic test set

(%)

Methods	Metamath Test Set	Synthetic Test Set		
		$n=3$	$n=4$	$n=5$
Holophrasm	67.0	73.8	69.3	27.4
MetaGen	80.0	73.9	69.5	64.4
GPT-4o	34.0	40.1	35.3	29.8
UnProver	**82.0**	**95.5**	**87.3**	**78.6**

注：n 表示运算符数量。

相比之下，UnProver 引入工作变量作为占位符，为推理过程提供明确的逻辑约束。与此同时，工作变量作为全局替换机制，每次模型输入均包含完整的子目标信息，使模型能够在全局范围内均衡考虑各子目标的可证明性。

此外，随着中间节点数量的增加，证明通过率呈下降趋势。这一现象与直觉一致：结构更复杂的表达式往往更难被证明。

5.3 消融实验

为系统评估数据增强策略与证明范式对模型性能的影响，本文设计了两组消融实验。第一组实验重点分析不同数据增强策略对模型泛化能力的提升效果；第二组实验则比较不同证明范式的建模方式，以验证本文所提出范式的高效性与证明能力。在 Metamath 测试集与合成测试集这两个评价场景下进行对比实验。

5.3.1 数据增强策略

定义 4 个用于模型微调的数据集。1）All：在原始数据集上应用所有数据增强策略。2）Subgoal：在原始数据集上仅使用子目标重排序策略。3）Subtree：在原始数据集上仅使用子树重建重构策略。4）Raw Dataset（UnProver）：从 set.mm 中提取的原始数据集。

如表 3 所列，在两个测试集上，UnProver 使用两种数据

增强策略的版本均取得最高的通过率，尤其是在合成测试集上，相较于基线模型有显著提升。结果表明，本文所提出的数据增强策略不仅有效增加了训练数据的多样性，而且显著提升了模型对未知结构的泛化能力，能够在此前未见过的定理上做出合理推断。

表 3 不同数据增强策略下的 Pass@1

Table 3 Pass@1 under different data augmentation strategies

(%)

Methods	Metamath Test Set	Synthetic Test Set		
		$n=3$	$n=4$	$n=5$
Raw Dataset	82.0	95.5	87.3	78.6
Subgoal	88.0	98.8	90.4	84.2
Subtree	93.0	99.1	92.4	87.3
All	**95.0**	**99.5**	**95.7**	**91.6**

5.3.2 证明范式

此外，本文对比了不同证明范式的建模方式：1) GPT-f，即 HTPS 所用的证明范式，模型需同时预测定理及其替换；2)本文方法，模型仅预测定理，而替换的生成由合一算法完成。

如表 4 所列，在两个测试集上，UnProver 的证明通过率均优于 GPT-f 。进一步地，统计 UnProver 与 GPT-f 的证明覆盖情况。两者共同证明的定理共有 44 个，GPT-f 独立证明的定理有 7 个，而 UnProver 独立证明的定理有 38 个，展现出 UnProver 的显著优势。此外，UnProver 在效率上同样表现突出：在相同的搜索资源下，其平均证明搜索时间比 GPT-f 缩短约 3 秒，且生成的证明平均长度减少接近 2 个证明步骤。这些结果表明，UnProver 不仅能够证明更多定理，而且能够以更高效、更简洁的方式完成证明。

表 4 不同证明范式上的 Pass@1 与平均证明时间

Table 4 Pass@1 and average proof time under different paradigms

Methods	Metamath Test Set/%	Synthetic Test Set/%			Average time /s
		$n=3$	$n=4$	$n=5$	
GPT-f	52.0	92.0	77.6	57.4	11.2
UnProver	**82.0**	**95.5**	**87.3**	**78.6**	**8.1**

5.4 实例分析

在 Metamath 测试集上使用 UnProver 进行搜索时，发现一些新的、更简洁的证明，这些证明已被官方正式接受。其证明长度缩短的原因大体可分为两类：

1)使用更高级的定理，这些定理比原始证明中使用的定理涵盖更多中间步骤；

2)采用新的证明策略。

图 7 展示了两类简化证明的示例，定理 uun111 的原始证明依赖于多个浅层定理的调用，而 UnProver 则预测更高级的定理 3anidm，一步即可完成推导(绿色背景)。从依赖拓扑上可见，该定理完整覆盖原始证明涉及的全部依赖，实现显著简化。定理 3o2cs 的原始证明需两次析取消解才能得出结论，而 UnProver 通过引入定理 3mix2，结合假设三段论的证明定理，仅用两步便实现了从假设到目标的链式蕴涵(绿色背景)，显著缩短了证明路径。

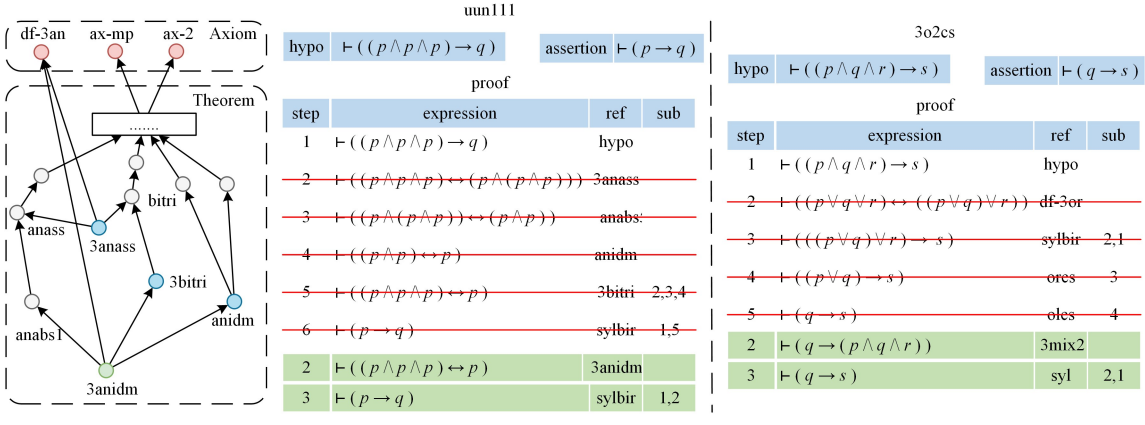

图 7 定理 uun111 和 3o2cs 的新证明及其引用关系拓扑图(电子版为彩图)

Fig. 7 New proofs of theorems uun111 and 3o2cs with reference topolog

UnProver 在失败情形下的表现，主要存在以下两类失败模式。

1)路径选择错误：模型未能选择正确的推理路径，而是在错误的分支上进行过多探索，导致搜索耗时过长并最终超时。这类错误通常源于模型对复杂结构的判断不足，使得启发式分值无法准确反映真实的推理价值。

2)搜索无法扩展：在部分证明中，搜索过程中的所有节点均无法扩展，导致搜索停滞。这种情况意味着模型未能生成具有足够覆盖性的候选定理，或搜索策略缺乏必要的回溯能力。

这些失败案例表明，尽管模型在大多数情况下能够指导搜索，但在路径探索效率与启发式选择上仍有改进空间。未来可考虑引入基于策略学习的全局搜索引导机制(如蒙特卡罗方法)，以缓解局部评分带来的偏差；同时通过扩大模型参数规模与训练数据规模，增强候选定理生成的多样性，从而提高搜索过程的鲁棒性与回溯能力。

5.5 结论

本文在 Metamath 形式化环境中开发 UnProver，其核心基于合一算法，通过在替换过程中无法立即推断的项上引入工作变量作为占位符，并利用合一算法来实例化这些工作变量。在此基础上，进一步提出数据增强策略。实验结果表明，UnProver 取得了更好的效果，而结合数据增强策略所训练的 UnProver 显著提升了模型的泛化能力，并且 UnProver 所发现的 6 个新的、更简洁的证明(3jaob1，bian1d2，2thALT，

orbi2iALT，pm3.48ALT，3jcadALT）获得了官方采纳。

尽管 UnProver 在结构化推理和泛化能力方面展现了明显优势，但仍存在若干局限：

1）Robinson 合一算法仅适用一阶逻辑，对高阶逻辑与类型系统无法保证完备性；

2）当前模型仍依赖启发式评分进行搜索，未结合更高级的策略优化。

结束语 在未来的工作中，UnProver 将并不限于命题逻辑，而是能够直接扩展至 Metamath 库中的各个领域。在当前实验中，由于研究重点是命题逻辑，因此所有工作变量均被指定为 WFFs 类型。对于 Metamath 中的其他分支，只需在该范式中引入相应的工作变量类型，即可有效指导相应的证明搜索。除此之外，未来还可进一步优化证明搜索算法、扩大模型规模与提升表达能力，并构建更大且更具多样性的训练数据，以增强系统在复杂推理场景下的鲁棒性与泛化能力。

参 考 文 献

[1] DE MOURA L，KONG S，AVIGAD J，et al. The Lean theorem prover（system description）[C]// Automated Deduction-CADE-25. Berlin：Springer，2015：378-388.

[2] COQ P. The Coq proof assistant-reference manual[R]. INRIA Rocquencourt and ENS Lyon，1996.

[3] NIPKOW T，WENZEL M，PAULSON L C. Isabelle/HOL：A proof assistant for higher-order logic[M]. Berlin：Springer，2002.

[4] POLU S，SUTSKEVER I. Generative language modeling for automated theorem proving[J]. arXiv：2009.03393，2020.

[5] MEGILL N，WHEELER D A. Metamath：A computer language for mathematical proofs[M]. Lulu. com，2019.

[6] LAMPLE G，LACROIX T，LACHAUX M A，et al. Hypertree proof search for neural theorem proving[C]// Advances in Neural Information Processing Systems 35. Curran Associates，2022：26337-26349.

[7] WANG M，DENG J. Learning to prove theorems by learning to generate theorems[C]// Advances in Neural Information Processing Systems 33. Curran Associates，2020：18146-18157.

[8] ROBINSON J A. A machine-oriented logic based on the resolution principle[J]. Journal of the ACM，1965，12(1)：23-41.

[9] BROWN T，MANN B，RYDER N，et al. Language models are few-shot learners[C]// Advances in Neural Information Processing Systems 33. Curran Associates，2020：1877-1901.

[10] WANG Y，LIU X，SHI S. Deep neural solver for math word problems[C]// Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. 2017：845-854.

[11] DUA D，WANG Y，DASIGI P，et al. DROP：A reading comprehension benchmark requiring discrete reasoning over paragraphs[J]. arXiv：1903.00161，2019.

[12] LI Z，KOVACHKI N，AZIZZADENESHELI K，et al. Neural operator：Graph kernel network for partial differential equations[J]. arXiv：2003.03485，2020.

[13] HAN J M，RUTE J，WU Y，et al. Proof artifact co-training for theorem proving with language models[J]. arXiv：2102.06203，2021.

[14] WANG H，YUAN Y，LIU Z，et al. Dt-solver：Automated theorem proving with dynamic-tree sampling guided by proof-level value function[C]// Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. 2023：12632-12646.

[15] POLU S，HAN J M，ZHENG K，et al. Formal mathematics statement curriculum learning[J]. arXiv：2202.01344，2022.

[16] YANG K，SWOPE A，GU A，et al. Leandojo：Theorem proving with retrieval-augmented language models[C]// Advances in Neural Information Processing Systems 36. Curran Associates，2023：21573-21612.

[17] XIN R，XI C，YANG J，et al. BFS-Prover：Scalable best-first tree search for LLM-based automatic theorem proving[J]. arXiv：2502.03438，2025.

[18] REN Z Z，SHAO Z，SONG J，et al. DeepSeek-Prover-V2：advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition[J]. arXiv：2504.21801，2025.

[19] LI Z，SUN J，MURPHY L，et al. A survey on deep learning for theorem proving[J]. arXiv：2404.09939，2024.

[20] WHALEN D. Holophrasm：a neural automated theorem prover for higher-order logic[J]. arXiv：1608.02644，2016.

[21] LIN X，CAO Q，HUANG Y，et al. ATG：Benchmarking automated theorem generation for generative language models[J]. arXiv：2405.06677，2024.

[22] ENDERTON H B. A mathematical introduction to logic[M]. San Diego：Elsevier，2001.

[23] AN C，CHEN Z，YE Q，et al. Learn from failure：Fine-tuning LLMs with trial-and-error data for intuitionistic propositional logic proving[J]. arXiv：2404.07382，2024.

[24] XUE L，BARUA A，CONSTANT N，et al. ByT5：Towards a token-free future with pre-trained byte-to-byte models[J]. Transactions of the Association for Computational Linguistics，2022，10：291-306.

[25] VASWANI A，SHAZEER N，PARMAR N，et al. Attention is all you need[C]// Advances in Neural Information Processing Systems 30. Curran Associates，2017.

[26] HURST A，LERER A，GOUCHER A P，et al. GPT-4o system card[J]. arXiv：2410.21276，2024.

CHEN Hongxiu，born in 2001，postgraduate，is a member of CCF（No. A12584G）. His main research interests include automatic theorem proving and natural language processing.

ZHAO Hengjun，born in 1985，Ph.D，associate professor，is a member of CCF（No. 50534M）. His main research interest is software theory and methods.

（责任编辑：柯颖）