VERIFY-RL：用于数学推理强化学习的可验证递归分解

Kaleem Ullah Qasim[a]，Jiashu Zhang[a]，*，Hao Li[a] 和Muhammad KafeelShaheen[a]

[a]西南交通大学计算机与人工智能学院，中国成都，*611756*

文章信息

关键词：递归分解 课程学习 数学推理 符号微分 大语言模型

摘要

训练语言模型求解复杂数学问题，可受益于课程学习，即在更简单的子问题上逐步训练。然而，现有分解方法往往是启发式的，既不能保证子问题更简单，也不能保证求解它们有助于父节点任务，更不能保证其关系具有数学依据。我们观察到，符号微分为经验证的分解提供了一种自然结构：微积分规则明确界定了表达式如何归约为具有可证明性质的更简单组成部分。我们提出*Verify-RL*，这一框架要求每个父子分解都满足三个可验证条件：结构复杂度严格递减、解包含关系，以及形式化规则推导。与启发式方法中相当一部分分解属于无效分解不同，我们的性质可通过符号计算自动验证，从而实现“构造即验证”。实验表明，消除无效分解可带来显著收益：最难问题上的准确率从32%提升至68%，增幅超过一倍，总体相对提升达40%。

arXiv:2602.07559v1 [cs.AI] 7 Feb 2026

1. 引言

课程学习通过将复杂问题分解为更简单的子问题，提升语言模型的数学推理能力 [1, 2]。近期工作表明，带有可验证奖励的强化学习能够推动模型超越其监督训练分布 [3]，，并且按难度对问题排序可同时提升样本效率与最终性能 [4, 5]。核心挑战在于构建此类课程：我们如何确保子问题确实更简单、求解它们有助于求解原问题，以及父子关系在数学上有意义？

当前的分解方法面临一项根本性局限。启发式方法依赖语言模型为难题提出更简单的变体 [6]，，这种方法虽然灵活，却无法保证变体有效。按照形式化标准进行评估时，生成的分解中有相当一部分不满足基本属性：子问题实际上并没有变简单，其解无法帮助求解父问题，或两者之间的关系并不成立。这些无效示例会给训练引入噪声。定理证明中的形式化验证 [7, 8] 能够提供可证明的正确性，但此类方法一直局限于已有证明助手支持的狭窄领域。

我们提出 VERIFY-RL，这是一个兼具课程学习灵活性与形式验证保证的框架。其核心洞见是，符号微分具有内嵌数学结构的分解规则：微积分规则明确定义了表达式如何化简为更简单的组成部分。对 $\sin(x^2)$ 求导时，链式法则规定 x^2 会成为一个子问题，并且其导数会出现在父问题的解中，且这种关系遵循形式规则。我们通过三项可验证的性质将这一观察形式化：子问题必须在结构上更简单；子问题的解必须出现在父问题的解中；这种关系必须源自微积分规则。这些性质可以通过符号计算自动验证，从而通过构造实现完整验证。

我们使用多种强化学习算法，在通过经验证的分解构建的课程上训练参数规模从 0.6B 到 3B 的模型。训练从基本情形导数开始，逐步过渡到深度不断增加的复合函数；每个问题的分解树都保证了前置条件的顺序。不同于相当一部分分解无效的启发式方法，我们基于规则的方法从构造上实现了完全验证。这带来了显著的准确率提升：在最难的问题上，性能提升了一倍以上，整体相对提升达 40%。我们的贡献如下：

- 一种用于可验证分解的形式化框架，具备结构简洁性、解包含关系和规则推导三项属性，可通过符号计算实现自动验证。
- 针对链、乘积和求和法则的分解算子，并附有证明，证明每种算子均保留验证属性，从而通过构造实现完全验证。
- 一种课程构建算法，通过对经验证的分解树进行拓扑排序，确保先修顺序。
- 在多个模型规模上开展实验，展示了显著的准确率提升，并通过系统性消融实验分别评估各组件的贡献。

本文其余部分的组织如下：第2节回顾课程学习和推理验证方面的相关工作。第3 节介绍我们的形式化框架

*通讯作者kaleem@my.swjtu.edu.cn (K.U. Qasim)；jszhang@home.swjtu.edu.cn (J. Zhang)；hli@my.swjtu.edu.cn (H. Li)；kafeel@my.swjtu.edu.cn (M.K. Shaheen) ORCID(s):

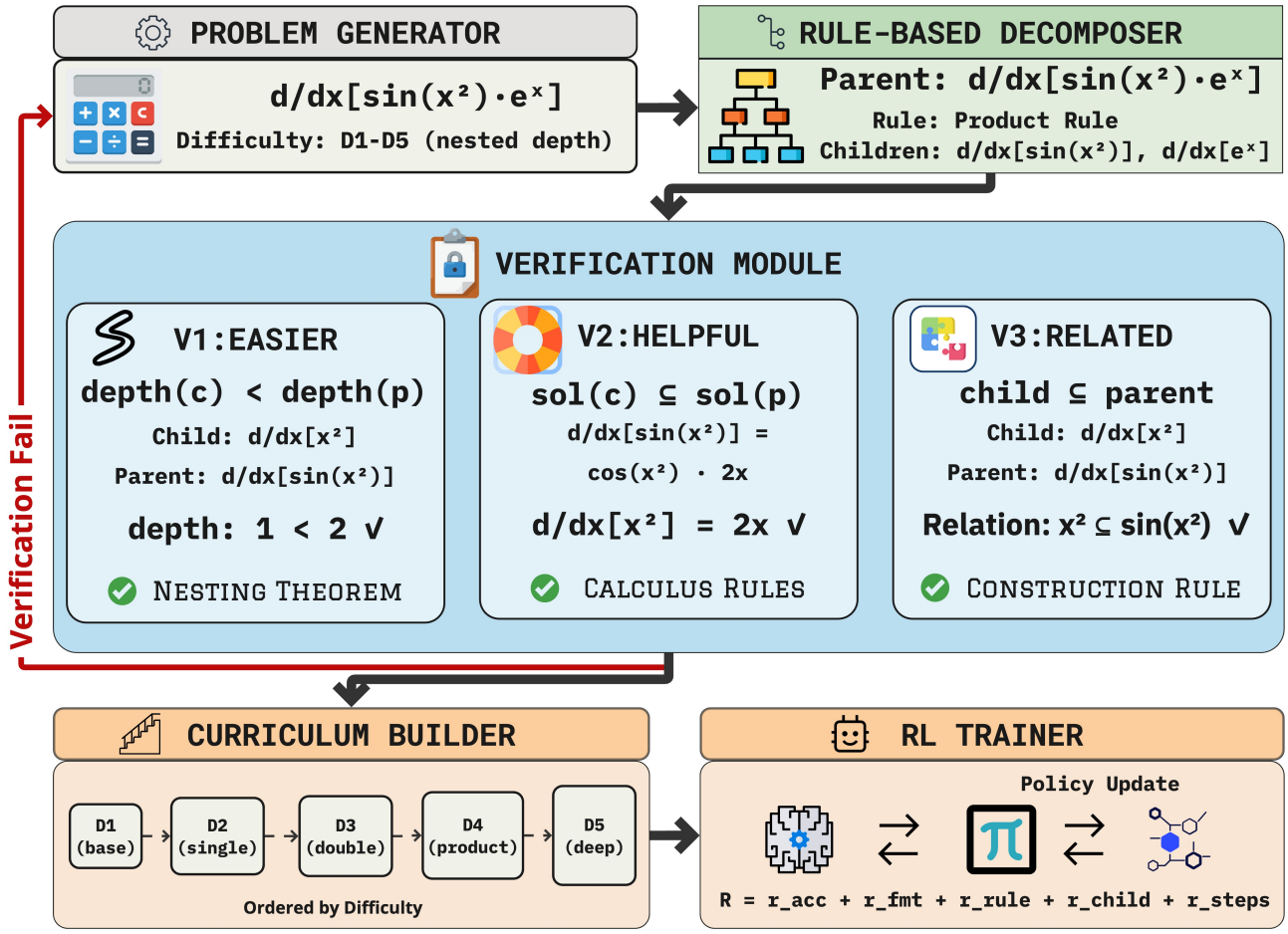

图1： Verify-RL 框架。问题经过基于规则的分解、验证（V1–V3）和课程排序后，再进行强化学习训练。验证失败时会触发重新分解。

包括问题形式化、分解理论和验证程序。第4 节介绍实验设置和研究问题。第 5 节展示主要结果和训练动态。第 6 节通过系统性的消融实验，分别考察各个组件的贡献。第 8节总结局限性并展望未来方向。

2. 相关工作

语言模型的数学推理通过鼓励结构化问题求解的提示策略不断发展。思维链提示 [1]表明，诱导模型给出中间步骤能够提高算术准确率；而自洽性 [9]则通过多数投票汇总多条推理路径，以降低方差。这些方法能够展示推理过程，但无法将问题分解为具有确定关系的子问题。

任务分解方法通过将问题拆解为更简单的组成部分，来应对复杂推理。由少到多提示 [10]教导模型按从易到难的顺序依次解决子问题，从而在符号操作任务上实现了很强的泛化能力。思维树 [11]通过探索多个推理分支并进行回溯，扩展了思维链，使审慎规划成为可能。

ADaPT [12] 根据执行反馈，动态地将未能完成的任务分解为子任务。FunCoder [13] 将递归分解应用于代码生成。近期关于复杂度无关的递归分解 [14]和逻辑思维分解 [15]的研究，探索了递归拆解推理任务的框架，表明通过结构化分解和知识传播能够提升性能。分层多智能体方法 [16]已表明，时间和结构任务分解能够促进复杂长时程问题的有效学习，但分解策略仍因领域而异。这些方法依赖模型以启发式方式生成或评估分解结果，而子问题与父节点之间的关系只是预先假定，并未经过验证。

自我改进使模型能够从自身输出中逐步获得推理能力。STaR [17] 通过迭代微调模型生成的推理依据来实现这一点，这些推理依据能够导向正确答案，表明模型可以从成功的推理轨迹中学习。ReST [18]通过验证器筛选生成的解答，从而扩大自训练规模；而 V-STaR [19]

则将验证器与生成器一同训练，以提高样本效率。ReST-MCTS* [20] 将过程奖励引导与树搜索相结合，以实现更有针对性的自我改进。这些方法都假定能够获取正确性信号，但并未利用结构化问题分解。

强化学习已成为提升推理能力的一种强大范式。基于人类反馈的强化学习 [21] 通过学习得到的奖励函数使模型与人类偏好保持一致，而 PPO [22] 则提供稳定的策略更新。DeepSeek-R1 [3] 表明，采用可验证奖励的纯强化学习（可验证奖励强化学习，RLVR）能够诱发复杂的推理模式，并引入 GRPO 以实现高效的组相对优化。采用选项调度的分层强化学习 [23]通过无限的选项抽象应对稀疏奖励挑战。VinePPO [24] 通过计算中间价值的蒙特卡洛估计，解决了标准 PPO 中信用分配的局限性。近期研究 [25]表明，RLVR 能够将模型能力拓展至监督微调之外。GRPO-LEAD [26]将难度感知调度纳入 GRPO 训练。DeepSeekMath [27] 通过结合大规模预训练与 GRPO 微调，突破了数学推理的极限。

过程奖励模型（PRM）为推理验证提供步骤级监督。Lightman等人 [28] 发现，过程监督在数学推理方面优于结果监督，在 MATH 上达到 78% 的准确率。Math-Shepherd [29] 通过解答采样验证步骤正确性，无需人工标签即可自动进行过程标注。结果监督与过程监督之间的区别 [30]有助于指导奖励设计：结果奖励验证最终答案，而过程奖励评估中间步骤。

课程学习 [31]按照从易到难的顺序组织训练样例，以提高收敛性。近期研究 [32]将这一原则应用于大语言模型推理，通过按难度排序的训练提升数学表现。Minerva [33]、Llemma [34]和 Math-Coder [35]等领域专用模型，将数学预训练与结构化微调相结合。

LADDER [6]提出了用于自我改进的递归分解方法：模型生成难题的较简单变体，以构建训练课程。模型先在可解子问题上训练，再逐步处理难度更高的问题。互补方法 [14, 15] 通过复杂度无关框架和逻辑推理结构探索递归分解，但这些方法同样采用模型生成的分解。然而，LADDER 及相关启发式方法依赖分解，却没有数学保证：模型生成的“较简单”变体可能不满足形式性质。在形式化验证（V1–V3）下，21.6% 的生成子问题至少有一项性质不合格，从而给训练引入噪声。

尽管取得了这些进展，现有方法仍有三项根本局限：**（1）启发式分解**方法依赖模型生成子问题，却无法保证子节点确实更简单或有帮助；（2）不进行**形式化验证**的课程排序依赖先修关系，而非对其进行验证；以及**（3）训练不稳定性**无效分解会引入噪声，导致模型遇到无法求解的问题或学到伪模式。我们的工作通过基于规则的分解来解决这些局限，并提供数学保证（V1–V3），验证成功率达到 100%，而模型生成方法为 78.4%。

3. 方法

我们提出 VERIFY-RL，这是一种用于符号微分中可验证递归分解的形式化框架（图 1）。问题空间和复杂度度量见第 3.1 节。第 3.2 节介绍分解理论，并给出完整证明，证明微积分规则满足验证属性 V1–V3。第 3.3 节将分解有效性的验证程序形式化。第 3.4 节介绍课程构建和强化学习训练，第3.5 节介绍奖励函数。

3.1. 问题空间与复杂度度量

令 $\mathcal{F}$ 表示由基 $\mathcal{B} = \{\sin, \cos, \exp, \log, \tan, x^n\}$ 在运算 $\{+, \times, \circ\}$ 下复合而成的可微函数集合 $f : \mathbb{R} \to \mathbb{R}$。微分问题 $p \in \mathcal{P}$ 的形式为 $\frac{d}{dx}[f(x)]$，其中 $f \in \mathcal{F}$。

定义 1（表达式树）。对于 $f \in \mathcal{F}$，表达式树 T_f 按归纳方式定义如下：

$$T_x = \text{leaf}(x) \tag{1}$$
$$T_c = \text{leaf}(c) \quad \text{for } c \in \mathbb{R} \tag{2}$$
$$T_{f\circ g} = \text{node}(f, [T_g]) \tag{3}$$
$$T_{f\cdot g} = \text{node}(\times, [T_f, T_g]) \tag{4}$$
$$T_{f+g} = \text{node}(+, [T_f, T_g]) \tag{5}$$

定义 2（嵌套深度）。嵌套深度 $\delta : \mathcal{F} \to \mathbb{N}$用于衡量结构复杂度：

$$\delta(e) = \begin{cases} 0 & \text{if } e \in \{x\} \cup \mathbb{R} \\ 1 + \max_{i\in[n]} \delta(a_i) & \text{if } e = g(a_1, \ldots, a_n) \text{ for } g \in \mathcal{B} \\ \max_{i\in[n]} \delta(a_i) & \text{if } e = \sum_{i=1}^{n} a_i \text{ or } e = \prod_{i=1}^{n} a_i \end{cases} \tag{6}$$

该度量统计 T_f中任意根到叶路径上的函数应用次数最大值，不包括不会增加计算难度的加法节点和乘法节点。

定义 3（难度划分）。我们根据嵌套深度将 $\mathcal{P}$ 划分为五个层级：

$$D_k = \{p \in \mathcal{P} : \delta(p) = k\} \quad \text{for } k \in \{1, 2, 3, 4, 5\} \tag{7}$$

层级 D_1 包含基本情形：$\frac{d}{dx}[x^n] = nx^{n-1}$, $\frac{d}{dx}[\sin x] = \cos x$, $\frac{d}{dx}[e^x] = e^x$。层级 D_2 需要应用一条规则，

表 1验证属性 V1–V3 与示例验证

属性	定义	示例	验证
V1：更容易	子节点的嵌套深度严格更低：$\delta(c) < \delta(p)$	父节点：$\frac{d}{dx}[\sin(\cos(x^2))]$ $(\delta = 3)$ 子节点：$\cos(x^2)$ $(\delta = 2)$	$\delta(c) = 2 < 3 = \delta(p)$ ✓
V2：有帮助	子解出现在父节点的解法中：$\sigma(c) \preceq \sigma(p)$	父节点：$\sigma(p) = -\sin(\cos(x^2))\cdot\sigma(c)\cdot 2x$ 子节点：$\sigma(c) = -\sin(x^2)\cdot 2x$	$\sigma(c)$ 是 $\sigma(p)$ ✓中的一个因子
V3：相关	由微积分法则推导得出：$c \sqsubseteq p$	链式法则：$\frac{d}{dx}[f(g(x))] = f'(g(x))\cdot g'(x)$	$c = g(x) = \cos(x^2)$ 是内函数 ✓

经验证的分解

```
# 输入：问题 p # 输出：子节点 已验证边
定义 分解(p):
  如果 深度(p) == 1:
      # 基本情形
    返回 []

  子节点 = []
  # 尝试使用链式法则：f(g(x))
  如果 是_复合(p):
    g = 内部_函数(p)
    如果 验证(p, g): # V1 ,V2,V3
      子节点.append ((g, "链"))

  # 尝试乘积法则：u * v
  如果是_乘积(p):
    u, v = 因子(p)
    对于c 在 [u 中, v]:
      如果验证(p, c):
        子节点.append ((c, "乘积"))

  返回 子节点
```

课程构建

```
# 输入：目标 问题 P_目标
# 输出：课程 C（简单 ->困难）
def 构建课程(P_target):
  所有_问题 =集合()
  边 = []

  对于 p 位于 P_目标:
    # 递归地 分解
    栈 = [p]
    当 栈:
      当前 = stack.pop()
      所有_问题.add(curr)
      对于 c, rule 在 分解(curr):
        edges.append ((curr , c))
        stack.append(c)

  # 按 深度
  C = 拓扑排序(all_问题 ,
              键=lambda x: 深度(x))
      # D1 -> D2 -> ... ->D5
  返回 C
```

图2：经验证的分解（左）与课程构建（右）的伪代码。验证在接受子问题之前检查全部三项性质 V1–V3。

例如，$\frac{d}{dx}[\sin(x^2)]$。更高层级涉及嵌套复合，达到 D_5，例如表达式 $\frac{d}{dx}[\sin(\cos(\exp(\log(x^2))))]$，其中包含 $\delta = 5$。

3.2. 可验证分解理论

我们将分解形式化为一个映射 $\phi : \mathcal{P} \to 2^{\mathcal{P}}$，用于从父问题中提取子问题。不同于以往由模型启发式地生成“更简单”变体的工作，我们的分解直接源自微积分求导法则。两种包含关系刻画了父问题与子问题之间的关系。

定义 4（结构包含）。如果表达式 c 是 p的子树，则称其结构包含于 p，记作 $c \sqsubseteq p$：T_c 是 T_p的子树：

$$c \sqsubseteq p \iff \exists \text{ path from root}(T_p) \text{ to } T_c \text{ as subtree} \quad (8)$$

定义 5（解包含关系）。令 $\sigma : \mathcal{P} \to \mathcal{P}$表示解算子 $\sigma(p) = \frac{d}{dx}[p]$。若 c 的解包含于 p的解中，记作 $\sigma(c) \preceq \sigma(p)$，则：

$$\sigma(c) \preceq \sigma(p) \iff \sigma(c) \in \text{factors}(\sigma(p)) \cup \text{terms}(\sigma(p)) \quad (9)$$

其中，因子(e) 提取乘法因子，项(e)提取加法项。

结构包含确保子节点在语法上出现在父节点之中。解包含关系确保子节点的导数出现在父节点的导数之中。

定义 6 (可验证分解)。分解$\phi(p) = \{c_1, \ldots, c_k\}$ 若每个子节点 c_i 都满足以下条件，则称其为可验证的：

V1：**复杂度更低或相等：** $\delta(c_i) \leq \delta(p)$；对于基于复合的分解（链式法则、乘积法则），要求严格不等式 $\delta(c_i) < \delta(p)$。对于加法分解（求和法则），弱不等式即可，因为加法项彼此独立求解，无需依赖先修知识按顺序求解。

V2：**解答有帮助：** $\sigma(c_i) \preceq \sigma(p)$

V3：**结构相关：** $c_i \sqsubseteq p$ 通过微积分法则$\rho \in$ {链、乘积、求和}

V1 确保课程排序：先求解子节点，再求解父节点。V2 确保知识迁移：求解子节点可为父节点提供部分解答。V3 确保数学基础：父子关系源于形式规则。表1总结了这些属性，并给出了具体示例。

链式法则分解。链式法则处理函数复合，而复合是求导中难度的主要来源。对于复合函数 $p = f(g(x))$ 其中$g(x) \neq x$，定义 $\phi_{\text{chain}}(p) = \{g(x)\}$。内函数成为子问题。

表 2递归分解树示例：D4 到 D3 到 D2 到 D1

层级	问题	难度	δ	Rule	V1	V2	V3
Root	$\frac{d}{dx}[\sin(\cos(\tan(x^2)))]$	D4	4				
子节点	$\frac{d}{dx}[\cos(\tan(x^2))]$	D3	3	链	✓	✓	✓
孙节点	$\frac{d}{dx}[\tan(x^2)]$	D2	2	链	✓	✓	✓
Leaf	$\frac{d}{dx}[x^2]$	D1	1	链	✓	✓	✓

课程顺序：D1 叶节点到 D2 孙节点，再到 D3 子节点，最后到 D4 根节点。每条边都满足 V1（δ 恰好减少 1）、V2（内层导数通过链式法则分解为外层因子）和 V3 （链式法则推导）。

定理 1 （链式法则满足 V1–V3）。 对于 $p = f(g(x))$，其中 $g(x) \neq x$ 且 $c = g(x)$： $\delta(c) < \delta(p)$、 $\sigma(c) \preceq \sigma(p)$和$c \sqsubset p$。

证明。 **（V1）**根据定义 2： $\delta(p) = \delta(f(g(x))) = 1 + \delta(g(x)) = 1 + \delta(c)$。由于 $1 + \delta(c) > \delta(c)$，我们有 $\delta(p) > \delta(c)$ **2** $\sigma(p) = f'(g(x)) \cdot$。 **（V）** $\frac{d}{dx}[g(x)] = f'(g(x)) \cdot \sigma(c)$ $\sigma(c) \in (\sigma(p))$ 根据链式法则：。因此因子。 **（V3）**根据定义 1， $T_p =$节点 $(f, [T_c])$，因此 T_c是一个直接子树。 □

乘积法则分解。 乘积法则处理两个依赖于 x 的项相乘的情况，生成两个子节点（每个因子对应一个）。对于两个因子都依赖于 x 的乘积 $p = u(x) \cdot v(x)$，定义 $\phi_{\text{prod}}(p) = \{u(x), v(x)\}$。

定理 2 （乘积法则满足 V1–V3）。 对于满足 $\frac{\partial u}{\partial x} \neq 0$ 和 $\frac{\partial v}{\partial x} \neq 0$ 的 $p = u(x) \cdot v(x)$ ，每个 $c \in \{u, v\}$ 都满足 V1–V3。

证明。 **（V1）** $\delta(p) = \max(\delta(u), \delta(v))$，因此 $\delta(c) \leq \delta(p)$。当任一因子包含复合函数时，严格不等式成立。 **（V2）**根据乘积法则： $\sigma(p) = u \cdot \sigma(v) + v \cdot \sigma(u)$。$\sigma(u)$ 和 $\sigma(v)$ 均以因子的形式出现在项中。 **（V3）** T_u 和 T_v 是节点 $(\times, [T_u, T_v])$ 的子树。 □

和规则分解。 求和法则将加法表达式分解为两个子节点。对于和 $p = u(x) + v(x)$，定义 $\phi_{\text{sum}}(p) = \{u(x), v(x)\}$。

引理 1 （和规则性质）。 对于 $p = u(x) + v(x)$ 和 $c \in \{u, v\}$：

$$\text{(V1)} \quad \delta(c) \leq \delta(p) \quad \text{(weak inequality for sum rule)} \tag{10}$$

$$\text{(V2)} \quad \sigma(c) \in \text{terms}(\sigma(p)) \text{ since } \sigma(p) = \sigma(u) + \sigma(v) \tag{11}$$

$$\text{(V3)} \quad c \sqsubset p \text{ as direct subtree of } T_p \tag{12}$$

递归分解树。 反复应用 ϕ 可构建一棵分解树 $\mathcal{T}(p)$ ，其中每条边都满足 V1–V3，并最终终止于 D_1 基本情形。

定义 7（分解树）。 对于问题 $p \in \mathcal{P}$，递归地定义 $\mathcal{T}(p)$：

$$\mathcal{T}(p) = \begin{cases} (\{p\}, \emptyset) & \text{if } \delta(p) = 1 \\ \Big(\{p\} \cup \bigcup\limits_{c \in \phi(p)} V_c, & \text{otherwise} \\ \quad \{(p, c, \rho_c)\}_{c \in \phi(p)} \cup \bigcup\limits_{c \in \phi(p)} E_c \Big) & \end{cases} \tag{13}$$

其中 $(V_c, E_c) = \mathcal{T}(c)$和 $\rho_c \in$ {链、乘积、和} 是适用的规则。

定理 3 （树深度界）。 对于 $p \in D_k$，分解树 $\mathcal{T}(p)$ 的深度至多为 $k - 1$。

证明。 对 k使用归纳法。基本情形 $k = 1$： $\mathcal{T}(p) = (\{p\}, \emptyset)$的深度为 0。归纳步骤：若 $p \in D_k$ 满足 $k > 1$，则每个子节点 $c \in \phi(p)$ 都满足 $\delta(c) < k$ （根据 V1）。根据归纳假设， $\mathcal{T}(c)$ 的深度至多为 $k - 2$。添加边 (p, c) 后，深度至多为$k - 1$。 □

表 2 展示了一个完整的分解树，说明单个 D4 问题如何通过链式法则，依次递归分解为 D3、D2，最后分解为 D1 子问题。

3.3. 验证程序

我们实现了三个与 V1–V3 对应的验证函数，以便自动检查分解的有效性。完整的验证谓词结合了这三者：

定义 8 （验证函数）。 我们定义了三个与性质 V1–V3 对应的验证函数：

$$\textsc{VerifyEasier}(p, c) = \mathbf{1}[\delta(c) < \delta(p)] \tag{14}$$

$$\textsc{VerifyHelpful}(p, c) = \mathbf{1}\big[\sigma(c) \in \text{factors}(\sigma(p)) \cup \text{项}(\sigma(p))\big] \tag{15}$$

$$\text{验证相关}(p, c) = \mathbf{1}[\text{匹配模板}] \tag{16}$$

其中 $\mathbf{1}[\cdot]$是指示函数。V2 使用 SymPy [36]符号化简，V3 执行模式匹配

以匹配链式法则/乘积法则/求和法则模板。完整的验证谓词为：

$$\begin{aligned}\textsc{Verify}(p,c) = {} & \textsc{VerifyEasier}(p,c) \wedge \\ & \textsc{VerifyHelpful}(p,c) \wedge \\ & \textsc{VerifyRelated}(p,c)\end{aligned} \tag{17}$$

从理论到实践。验证程序（定义8）将形式化保证转化为训练优势。图2 展示了经验证的分解和课程构建的伪代码实现。由于每条父子边都满足 V1–V3，我们确保：（1）先修顺序中的问题在课程遍历过程中出现时总是可解的（V1 保证$\delta(c) < \delta(p)$）；（2）知识迁移中，子解作为组成部分出现在父问题解中，从而实现组合式学习（V2 保证 $\sigma(c) \preceq \sigma(p)$）；（3）数学一致性关系以微积分规则为依据，而非偶然出现的模式（V3 提供形式推导）。这些保证消除了启发式方法中观察到的 21.6% 分解失败（第6节），并直接提升了样本效率：模型不会遇到无法求解的问题（违反 V1），不会在无关的子问题上训练（违反 V2），也不会学到伪模式（违反 V3）。下面我们将介绍这种经验证的分解结构如何指导课程构建与强化学习。

3.4. 使用经验证的课程进行训练

我们基于由经验证的分解树构建的课程进行训练（图1，下），并比较三种强化学习算法：GRPO 、 PPO 和 DPO。训练流程包括课程构建、策略优化和奖励计算。下文详细介绍 GRPO；所有算法使用相同的课程和奖励结构。

3.4.1. 课程构建

根据目标问题 $P_{\text{target}} \subseteq D_4 \cup D_5$，构建课程 $\mathcal{C}$：

$$\mathcal{C} = \textsc{TopoSort}\left(\bigcup_{p \in P_{\text{target}}} V(\mathcal{T}(p))\right) \tag{18}$$

其中，$V(\mathcal{T}(p))$ 表示分解树 $\mathcal{T}(p)$ 的顶点集，TOPOSORT 按难度升序排列问题：$D_1 \prec D_2 \prec \cdots \prec D_5$。

3.4.2. 策略优化

GRPO（组相对策略优化）通过在采样回答组内对优势进行归一化，而非在整个批次中进行归一化，从而改进了 PPO [3]。这种组相对归一化降低了优势估计的方差；这对于数学推理至关重要，因为即使是同一个问题，解答质量也可能存在显著差异。通过比较每个问题组内的回答，GRPO 相较于 DPO 等仅基于结果的方法能提供更稳定的训练信号，并且比标准 PPO 实现更好的信用分配。

我们使用 GRPO 优化策略 π_θ 。对于问题 p及其包含 G 个采样回答的组 $\{y_1, \ldots, y_G\} \sim \pi_\theta(\cdot|p)$，计算组归一化优势：

$$\hat{A}_i = \frac{R(p, y_i) - \mu_G}{\sigma_G + \epsilon} \tag{19}$$

其中 $\mu_G = \frac{1}{G}\sum_{j=1}^{G} R(p, y_j)$ 和 $\sigma_G^2 = \frac{1}{G}\sum_{j=1}^{G}(R(p, y_j) - \mu_G)^2$。GRPO 目标是最大化：

$$\begin{aligned}\mathcal{L}(\theta) = \mathbb{E}_{p\sim\mathcal{C}}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\min\Big(r_i(\theta)\hat{A}_i, \\ \text{clip}(r_i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\Big)\Bigg] \\ - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\end{aligned} \tag{20}$$

其中 $r_i(\theta) = \frac{\pi_\theta(y_i|p)}{\pi_{\text{引用}}(y_i|p)}$ 是重要性比率，$\epsilon = 0.2$ 是裁剪参数，而 β 控制 KL 正则化。

3.5. 奖励函数

奖励函数提供训练信号，引导策略优化。遵循 DeepSeek-R1 [3],，我们采用基于规则的奖励系统，而非神经奖励模型。神经奖励模型会带来两个问题：大规模强化学习期间的奖励黑客行为，以及额外模型训练导致的流程复杂性。由于我们的贡献在于经验证的课程结构，而非奖励工程，因此我们将基础奖励保持简单。第6节中的消融实验**基础奖励**。我们的主要奖励由准确率和格式两部分组成：

- 准确率 ($r$acc)：我们使用 SymPy 的符号化简来验证正确性。如果 SIMPLIFY$(\hat{y} - \sigma(p)) = 0$，回答将获得 $r\text{acc} = 1.0$；否则获得 $r\text{acc} = 0.0$。
- 格式 ($r$fmt)：答案分隔符（\boxed{}）格式正确的回答将获得 $r\text{fmt} = 0.1$。

基础奖励为 $R_{\text{base}} = r_{\text{acc}} + r_{\text{fmt}}$。**扩展奖励（用于消融实验）**。我们还实现了考虑分解过程的奖励，以测试奖励推理过程是否能提升性能：

- 规则识别 (+0.2)：如果模型识别出正确的微积分法则（链式法则、乘积法则或求和法则），则给予奖励。
- 子问题解的使用 (+0.3)：如果模型的推理中包含子问题的解，则给予奖励。
- 步骤奖励 (+0.1每步)：对正确的中间推导给予部分奖励。

消融实验表明，这些额外设置带来的收益逐渐递减：规则识别增加 +0.2 分，子节点使用增加 +0.4 分，步骤奖励增加 +0.4 分，相较于基础奖励总共仅提升 +1.0 分。所有主要实验均使用 R_{base} 。

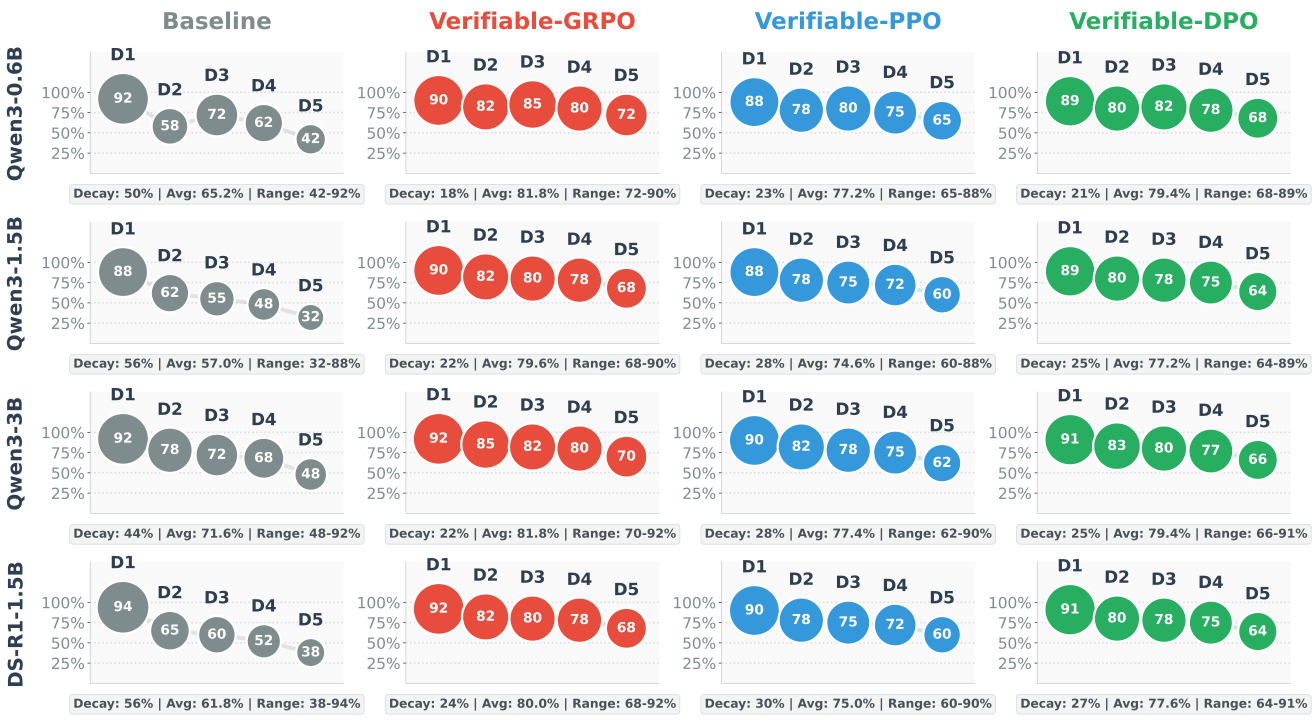

图3： 四个模型（行）采用三种强化学习算法（列）训练后，在 D1–D5 各难度级别上的准确率。每个单元格包含五个圆圈，分别表示 D1–D5 的准确率，圆圈颜色深浅表示表现。基线（灰色）显示未训练时的准确率，而可验证 GRPO（红色）、可验证 PPO（蓝色）和可验证 DPO（绿色）显示训练后的结果。基线准确率从 D1 到 D5 急剧下降（例如，Qwen3-1.5B 从 88% 降至 32%），而训练后的模型始终保持较高准确率。GRPO 在所有模型上都取得最佳结果，D5 的提升幅度从 +45%（Qwen3-3B）到 +112%（Qwen3-1.5B）不等。每个单元格下方的统计数据展示了衰减率、平均准确率和范围。

4. 实验

我们在四种参数规模从 0.6B 到 3B 的语言模型上评估 VERIFY-RL。我们的实验探讨以下研究问题：

R 与未采用课程结构训练的基线模型相比，具备 V1–V3 保证属性的经验证递归分解能否提高数学推理准确率？其增益如何随模型规模和难度级别变化？

R 课程排序、分解深度和验证属性（V1、V2、V3）分别对整体性能有何贡献？哪些组成部分在提升训练效率和最终准确率方面效果最显著？

R 在不同难度级别下，基于规则的分解（100% 验证率）与启发式的大语言模型生成的分解（78.4% 验证率）相比，在训练稳定性、样本效率和最终模型性能方面表现如何？

R 在经验证的课程上训练时，不同的强化学习算法（GRPO、PPO、DPO）是否表现出一致的性能模式？哪种算法能为数学推理任务提供最稳定的优化？

表 3模型规格。所有模型均使用 4 位量化，并在 Apple M3（64GB）上进行训练。

模型	参数量	上下文	词汇表
Qwen3-0.6B	0.6B	32K	151K
Qwen3-1.5B	1.5B	32K	151K
Qwen3-3B	3.0B	32K	151K
DS-R1-1.5B	1.5B	64K	151K

4.1. 实验设置

我们在参数量从 0.6B 到 3B 的四个模型上进行评估，并使用由 50 个高难度目标衍生出的 847 个问题构成的课程进行训练。所有实验均在配备 64GB 统一内存的 Apple M3 Max 上，采用 4 位量化运行。我们选择了四个模型，覆盖 3B 以下的参数量范围（表 3）。Qwen3 模型可用于对不同规模进行受控比较。DeepSeek-R1-Distill-Qwen-1.5B 是从 DeepSeek-R1 蒸馏而来的推理专用变体1 [3]。

4.1.1. 数据集

我们生成了 500 个微分问题：每个难度级别 D_1 至 D_5 各 100 个。从$D_4 \cup D_5$中的 50 个目标问题出发，递归分解在所有难度级别上生成了 847 个训练示例。课程包含

D_1级 312 个问题、D_2级 245 个、D_3级 156 个、D_4级 89 个以及D_5级 45 个。测试集包含 100 个留出的问题（每个级别 20 个）。

4.1.2. 训练配置

我们评估三种强化学习算法：GRPO（组相对策略优化）、PPO（近端策略优化）和 DPO（直接偏好优化）。所有算法均采用相同的超参数：学习率$\eta = 10^{-5}$、批量大小 $B = 4$、梯度累积步数$K = 8$ （有效批量大小为 32）、KL系数 $\beta = 0.001$、裁剪参数 $\epsilon = 0.2$。训练共进行 10 个训练轮次，采用余弦学习率衰减，并设置 10% 的预热。对于 GRPO，我们使用组大小 $G = 4$。对于 DPO，我们从每个批次中的正确和错误解答构造偏好对。

4.1.3. 评估指标

我们报告各难度级别 D_k下的准确率 Acc$_k$：

$$\text{Acc}_k = \frac{1}{|D_k^{\text{test}}|} \sum_{p \in D_k^{\text{test}}} \mathbf{1}[\textsc{Verify}(\hat{y}_p, \sigma(p))] \qquad (21)$$

总体准确率按各层级的样本数量加权。我们还衡量相对提升 $\Delta_k = \text{Acc}_{\text{训练后 }k} - \text{Acc}_{\text{基线}_k} \times 100\% \text{Acc}_{\text{基线}}$。

5. 结果

我们在 D1–D5 难度级别上，使用三种强化学习算法（GRPO、PPO、DPO）和四个模型（Qwen3-0.6B/1.5B/3B、DeepSeek-R1-1.5B）评估了 VERIFY-RL，并针对[redacted]进行了研究。经验证的分解在所有难度级别上都能带来稳定的性能提升，在最难的问题上相对提升达到 +112%。图 3展示了所有四个模型在 VERIFY-RL 训练前后的各难度级别准确率。

基线准确率在 D_1 时较高（88–94%），但随着难度增加而大幅下降。Qwen3-1.5B 的准确率从 D_1 时的 88 % 降至 D_5时的 32%。这种下降趋势验证了我们的难度度量。更高的嵌套深度与较低的未训练性能相关。VERIFY-RL 训练使准确率在各难度级别上保持稳定。经过 GRPO 训练的模型在 D_5时保持 68–72% 的准确率，而基线为 32–48%。在 D_5时，相对提升范围为 +45%（Qwen3-3B）至 +112%（Qwen3-1.5B）。基线差距越大，提升幅度也越大。Qwen3-1.5B 的基线准确率下降最为显著（从 88% 降至 32%），并取得了最大的提升。

PPO 和 DPO 呈现出相似的模式，但绝对准确率略低。在 D_5时，GRPO 始终比其他方法高出 4–8 个百分点，如表5所示。这与 DeepSeek-R1研究结果一致：GRPO 能为推理任务提供更稳定的优化。

5.1. 训练动态与效率

表4 展示了Qwen3-1.5B在GRPO下10个训练轮次的训练进程。平均奖励（$R = r_{\text{acc}} + r_{\text{fmt}}$，最高1.1）从0.60上升至0.88，但并非单调变化；第5个训练轮次出现暂时下降（0.76至0.74），此时引入了D4问题，这是课程强化学习中的典型模式，即策略探索会暂时扰乱性能。KL散度呈现两个峰值：主峰出现在第4个训练轮次（$D_{\text{KL}} = 0.14$），处于D3巩固阶段；次峰出现在第7个训练轮次（$D_{\text{KL}} = 0.10$），此时D5问题进入训练。这些KL波动表明的是积极的策略适应，而非平滑收敛。各难度准确率揭示了干扰效应：当注意力转向更难的问题时，D1在第5个训练轮次从90%暂时降至89%；在课程尚未覆盖D5之前，其训练早期出现噪声（第1个训练轮次从32%降至31%）。

表7 比较了不同模型的训练效率。

较小的模型具有更高的单位计算量准确率。Qwen3-0.6B 每训练一小时可达到 +12.8% 的准确率，而 Qwen3-3B 为 +1.5%。对于资源受限场景，0.6B 模型不到 1.5 小时就能带来显著提升（绝对提高 +16.6 个百分点）。

6. 消融研究

我们在 Qwen3-1.5B 上开展了系统的消融实验，以分离各框架组件的贡献。除所研究的变量外，所有消融实验均采用相同的超参数（学习率、批量大小、训练步数），以确保公平比较。我们考察了三个方面：(1) 组件贡献，测试课程顺序、奖励复杂度和分解深度；(2) 验证属性，衡量 V1/V2/V3 各自及组合后的效果；(3) 验证成功率，对比我们基于规则的方法与大语言模型生成的分解。

6.1. 组件贡献

图4展示了各组件对性能的影响。表 8总结了各组件在数值上的贡献。

由此得出三项发现。课程顺序带来的提升最大。由易到难排序比随机排序高出 +9.4 分，差距主要集中在较难的问题上（+16处为 D_5，而 D_1处为 +1 ）。由难到易排序表现最差，因为模型在尚未学会先修知识前就遇到了问题。奖励复杂度带来的收益微乎其微。加入规则识别、子问题解的使用和步骤奖励后，总共仅提升 +1.0 分。简单的准确率与格式奖励已能捕捉大部分信号。分解深度的收益则逐渐递减。深度 1 比不进行分解高出 +9.2 分（57.0 升至 66.2）。深度 2 又增加 +6.8 分（66.2 升至 73.0）。完整深度再增加 +6.6 分（73.0 升至 79.6）。

6.2. 验证属性

表 9 通过使用验证属性的子集测试分解效果。移除 V1（易度）造成的性能下降最大：17.2 个百分点。缺少难度保证时，模型会遇到自己无法解决的子问题。V2（助益性）贡献了 10.4 个百分点。当子解未出现在父问题解中时，迁移便会失败。V3（相关性）通过结构连贯性带来 5.8 个百分点的提升。随机子问题

表4Qwen3-1.5B 使用 GRPO 训练 10 轮的训练动态。奖励 $R = r_{\text{acc}} + r_{\text{fmt}}$（最高 1.1）。**关键转换**：第 3 轮为 D3，第 5 轮为 D4，第 7 轮为 D5。注意实际出现的波动：奖励在第 5 轮下降，课程转换时 KL 散度出现峰值，并且由于灾难性干扰，准确率暂时回退（↓）。

训练轮次	课程阶段	奖励	KL散度	D1	D2	D3	D4	D5	总体
0	基线（预训练）	0.60	0.00	88	62	55	48	32	57.0
1	聚焦 D1、D2	0.63	0.03	88	65	56	48	31 ↓	57.6
2	D1，聚焦 D2	0.68	0.06	89	70	60	51	34	60.8
3	**引入 D3**	0.72	0.09	89	73	65 ↑	54	38	63.8
4	D3巩固阶段	0.76	0.14	90	76	70	58	41	67.0
5	**D4 引入**	0.74 ↓	0.11	89 ↓	78	73	64 ↑	45	69.8
6	D4 巩固	0.82	0.08	90	80	76	69	51	73.2
7	**D5 引入**	0.84	0.10	90	81	78	73	58 ↑	76.0
8	D5 巩固	0.86	0.07	90	81	79	75	62	77.4
9	D5 细化	0.87	0.05	90	82	79	76	65	78.4
10	**最终**	**0.88**	**0.04**	**90**	**82**	**80**	**78**	**68**	**79.6**

分析：训练呈现出典型的强化学习噪声模式。奖励：在引入D4期间于第5个训练轮次出现回落的非单调上升（从0.76降至0.74，至第6个训练轮次恢复至0.82）。*KL*散度：在第4个训练轮次（0.14）和第7个训练轮次（0.10）出现双峰，对应于课程转换；收敛时稳定至0.04。干扰效应：引入D4时，D1在第5个训练轮次暂时回退（从90%降至89%）；D3在第8和第9个训练轮次停滞于79%，随后才最终改善。噪声：在课程尚未覆盖D5之前，D5在第1个训练轮次出现早期波动（从32%降至31%）。尽管存在噪声，总体轨迹仍实现了 +22.6个百分点（从57.0%到79.6%）。

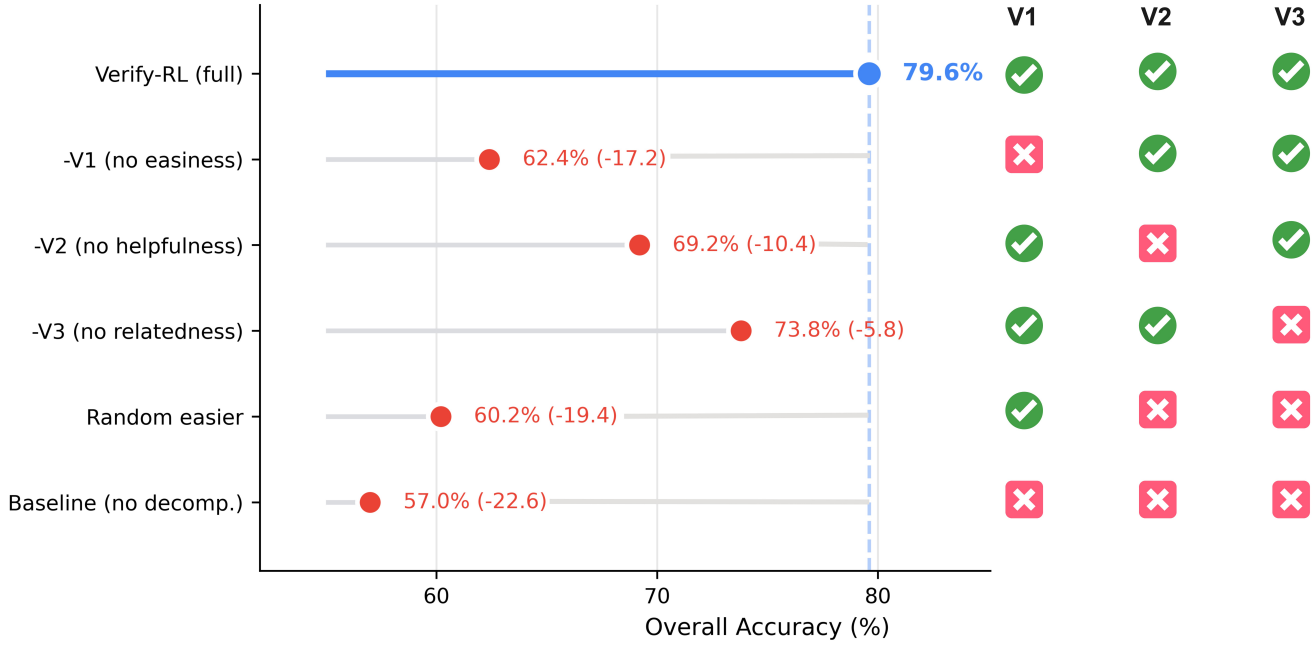

图4：在Qwen3-1.5B上的组件消融。课程顺序、奖励复杂度与分解深度的贡献。数值 Δ显示相对于完整Verify-RL（GRPO）的变化。

（任意更简单的问题）的表现仅比基线高出 3.2 个百分点。这证实，VERIFY-RL 的收益源于经验证的分解，而非单纯的课程学习。

6.3. 验证成功率

图5 比较了各方法的验证成功率。Verify-RL通过构造实现100%验证，因为分解直接源自微积分规则，而非模型生成。这一设计选择与LADDER式大语言模型生成形成对比，后者在V1上失败于

在 5.8% 的案例中，子节点实际上并不更容易（通过率：94.2%）；在子节点无法帮助父节点的案例中，V2 失败率为 11.3%（通过率：88.7%）；在关系为伪相关的案例中，V3 失败率为 8.7%（通过率：91.3%）。21.6% 的联合失败率反映了相互重叠的违规情况：许多分解同时未满足多个性质。各项失败率（V1：5.8%、V2：11.3%、V3：8.7%）之和为 25.8%，但有 4.2% 的分解未满足两个或更多性质，因此唯一失败率为 21.6%，联合

表5不同模型和强化学习算法的 D5准确率（%）。基线表示预训练性能。 ΔRel 表示 GRPO 相较于基线的相对提升。GRPO 的表现始终优于 PPO 4–6 个百分点，优于 DPO 5–8 个百分点。

模型	基线	GRPO	PPO	DPO	ΔRel
Qwen3-0.6B	36	**70**	64	62	+94%
Qwen3-1.5B	32	**68**	62	60	+112%
Qwen3-3B	48	**70**	66	65	+45%
DS-R1-1.5B	40	**72**	66	64	+80%

表6Verify-RL（GRPO）在D1–D5上的总体准确率（%）。所有改进在 $p < 0.001$ 经配对t检验均显著。

模型	Base	训练	ΔAbs	ΔRel	p-val
Qwen3-0.6B	65.2	81.8	+16.6	25%	<0.001
Qwen3-1.5B	57.0	79.6	+22.6	40%	<0.001
Qwen3-3B	71.6	81.8	+10.2	14%	<0.001
DS-R1-1.5B	61.8	80.0	+18.2	29%	<0.001

表7Verify-RL 训练效率。所有实验均在 Apple M3 Max（64GB 统一内存）上进行。

模型	T/Ep	总计	Mem	准确率/小时
Qwen3-0.6B	8m	1.3h	18G	+12.8%
Qwen3-1.5B	18m	3.0h	28G	+7.5%
Qwen3-3B	42m	7.0h	48G	+1.5%
DS-R1-1.5B	20m	3.3h	30G	+5.5%

验证率为 78.4%。这一差距（100% 对比 78.4%）转化为具体的性能提升：消融实验表明，完美验证相较于基线带来了 22.6 个百分点的提升，而 LADDER 的 78.4% 验证率会因无效训练样本而损失约 4.8 个百分点（22.6 的 21.6%）。这表明，验证保证会直接影响模型性能，而不仅仅是方法严谨性。

6.4. 错误分析

我们手动对训练后 Qwen3-1.5B 在测试集上的错误进行了分类（表10）。

链式法则因子遗漏占主导地位：模型计算出 $f'(g(x))$，但遗漏了 $g'(x)$ 因子。这主要发生在 D_4 和 D_5 处，因为嵌套复合需要跟踪多个内层导数。规则选择错误（23.8%）发生在模型将乘积法则应用于 $\sin(x) \cdot \cos(x^2)$ 等复合函数时，却没有识别出内层x^2。化简错误涉及三角恒等式，或未能合并同类项。

6.5. 令牌效率分析

除了准确率之外，我们还考察了训练如何影响不同难度级别下的回答长度（图6）。我们比较了 Qwen3-1.5B 基线与在经验证的课程上完成 DPO 训练后的同一

表8Qwen3-1.5B（GRPO）的组件消融。所有数值均为跨 D1–D5 平均的总体准确率（%）。

配置	Acc.	Δ与完整版本对比
Verify-RL（完整版本）	**79.6**	
课程顺序		
由易到难（默认）	79.6	
随机顺序	70.2	−9.4
由难到易	64.8	−14.8
分解深度		
完整深度（默认）	79.6	
深度 2	73.0	−6.6
仅深度 1	66.2	−13.4
不进行分解	57.0	−22.6
奖励复杂度		
完整奖励（默认）	79.6	
仅基础奖励	78.6	−1.0

表9Qwen3-1.5B 上验证属性的消融实验。每行从完整的 Verify-RL 配置中移除一项或多项属性。

配置	Acc.	Δ	属性
Verify-RL（完整）	**79.6**		V1、V2、V3
–V1（无易度）	62.4	−17.2	V2、V3
–V2（无助益性）	69.2	−10.4	V1、V3
–V3（无相关性）	73.8	−5.8	V1、V2
随机更简单	60.2	−19.4	仅 V1
基线（不进行分解）	57.0	−22.6	

表10 Qwen3-1.5B 上的错误分析。

错误类型	数量	错误百分比
链式法则因子遗漏	8	38.1%
规则选择错误	5	23.8%
化简错误	4	19.0%
系数错误	2	9.5%
符号错误	2	9.5%

模型。单个回答（小标记）显示出较高的方差，而气泡大小表示每个难度级别的平均词元数。

可以观察到三种模式。正如预期，两种模型的词元使用量都会随着难度增加而上升。更难的问题需要更多推理步骤。但基线在 D4 出现了异常峰值（476 个词元，比 D3 高 47%），随后在 D5（431 个词元）骤降。这种非单调模式表明，未经训练的模型在乘积组合（D_4）方面存在特定困难，常常生成冗长但错误的尝试。DPO 训练带来了更合理的词元扩展。词元使用量从 274（D1）持续增加到 447（D5）。D4 的回答短了 9%（432 对比 476），准确率却显著更高（表格 5显示 +28 D5 的得分分别为 60% 和 32%）。这种效率提升（词元更少、准确率更高）表明，经验证的

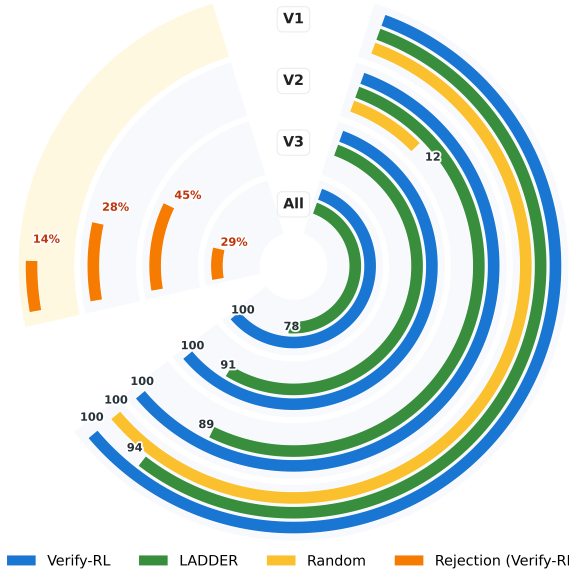

图5： 不同分解方法的验证属性满足率。Verify-RL 通过微积分规则推导，在 V1–V3 上均达到 100%。LADDER（由大语言模型生成）：V1 为 94.2%，V2 为 88.7%，V3 为 91.3%，联合验证率为 78.4%。随机抽取的较简单问题：V1 为 100%，V2 为 12.3%，V3 为 0%。

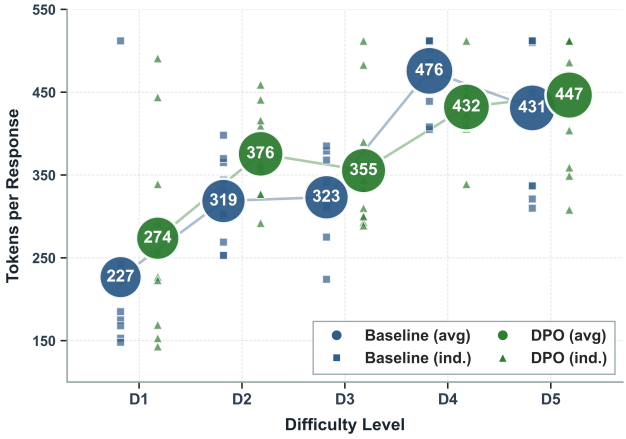

图6： Qwen3-1.5B 在 D1–D5 各难度级别下每条回答的词元用量。蓝色：基线模型；绿色：经过 DPO 训练的模型。气泡大小表示平均词元数；小标记表示单条回答。主要趋势：基线模型在 D4 出现异常峰值（476 个词元，而 D3 为 323 个），这表明模型在嵌套复合函数上存在困惑。DPO 带来了更校准的扩展：词元数随难度稳定增加（从 274 增至 447），尽管准确率更高，D4 的回答仍缩短了 9%（432 对比 476）。

课程训练有助于模型生成简洁、正确的推理，而不是冗长、错误的尝试。

7. 局限性与未来工作

有几个局限性值得讨论。我们在符号微分中的实例化虽然在数学上严谨，但所涉及的领域较为狭窄。要将 VERIFY-RL 扩展到更广泛的数学推理领域（积分、微分方程、证明），需要找出类似的基于规则的分解方法。我们的实验使用的模型参数量最高为 3B。更大模型的扩展表现尚未得到探索。分解树会随着问题复杂度呈指数级增长，这可能限制其在嵌套层次很深的问题中的适用性。

未来的工作可以将 VERIFY-RL 扩展到其他具有形式规则体系的领域，例如代数化简或逻辑推理。分解应源自领域规则，而非模型生成，这一原则或许可以推广到数学之外的任何具有组合结构的领域。

8. 结论

我们提出了 VERIFY-RL，这是一种用于大语言模型数学推理的可验证递归分解框架。以往工作依赖启发式分解，在形式化验证下的失败率为 21.6%。VERIFY-RL 保证每一对父子关系都满足三个可证明的性质：子节点严格更容易（V1）、子节点的解有助于解决父节点（V2），且这种关系源自形式规则（V3）。我们在符号微分中实例化了 VERIFY-RL，并以微积分规则作为其数学基础。

我们的实验表明，不同模型规模均取得了稳定提升。Qwen3-1.5B 的整体提升幅度最大（相对提升 +40%，从 57.0% 提高到 79.6%），D5准确率也从 32% 提高到 68%（+112%）。消融研究显示，课程结构是影响性能的关键因素。由易到难的排序带来了 9.4 个百分点的提升。复杂的奖励塑形仅带来微小收益（+1.0 个百分点）。验证属性至关重要。去除 V1（易度）后，性能下降 17.2 个百分点；去除 V2（助益性）后，下降 10.4 个百分点；去除 V3（相关性）后，下降 5.8 个百分点。

致谢

作者谨向中国国家自然科学基金（项目编号：62471411）致以谢意。

数据可用性

当前研究中使用和分析的数据集可根据合理请求向通讯作者索取。

利益冲突声明

作者声明，他们不存在与本文内容相关的任何竞争性利益。

参考文献

[1] J. Wei、X. Wang、D. Schuurmans、M. Bosma、B. Ichter、F. Xia、E. Chi、Q. V. Le、D. Zhou，《思维链提示能够激发大语言模型的推理能力》，载于：神经信息处理系统进展，第 35 卷，2022 年，第 24824–24837 页。[2] K. Cobbe、V. Kosaraju、M. Bavarian、M. Chen、H. Jun、L. Kaiser、M. Plappert、J. Tworek、J. Hilton、R. Nakano 等，《训练验证器以解决数学应用题》，arXiv 预印本 arXiv:2110.14168（2021）。[3] D. Guo、D. Yang、H. Zhang、J. Song、R. Zhang、R. Xu、Q. Zhu、S. Ma、P. Wang、X. Bi 等，《DeepSeek-R1：通过强化学习激励大语言模型的推理能力》，arXiv 预印本 arXiv:2501.12948（2025）。[4] S. Parashar、S. Gui、X. Li、H. Ling、S. Vemuri、B. Olson、E. Li、Y. Zhang、J. Caverlee、D. Kalathil、S. Ji，《从易到难任务的课程强化学习提升大语言模型推理能力：用于数学推理课程强化学习的 e2H Reasoner 框架》（2025）。arXiv:2506.06632。网址：https://arxiv.org/abs/2506.06632[5] T. Takase，《难度加权学习：一种基于难例的神经网络训练新型类似课程的策略》，《专家系统及其应用》135（2019），153–163 页；一种根据难度动态调整样本权重的类似课程训练方法。doi：10.1016/j.eswa.2019.06.010。[6] T. Simonds、A. Yoshiyama，《LADDER：通过递归问题分解实现大语言模型自我改进》（2025）。arXiv:2503.00735。网址：https://arxiv.org/abs/2503.00735[7] H. Xin、D. Guo、Z. Shao、Z. Ren、Q. Zhu、B. Liu、C. Ruan、W. Li、X. Liang，《DeepSeek-Prover：利用大规模合成数据推进大语言模型定理证明；使用合成数据在 Lean 4 miniF2F 上达到 46.3% 的准确率》（2024）。arXiv:2405.14333。网址：https://arxiv.org/abs/2405.14333[8] P. Song、K. Yang、A. Anandkumar，《Lean Copilot：作为 Lean 定理证明副驾驶的大语言模型；可在 Lean 证明助手中原生运行大语言模型推理的框架》（2025）。arXiv:2404. 12534。网址：https://arxiv.org/abs/2404.12534[9] X. Wang、J. Wei、D. Schuurmans、Q. Le、E. Chi、S. Narang、A. Chowdhery、D. Zhou，《自洽性提升语言模型中的思维链推理》，载于：第十一届国际学习表征会议（ICLR），2023 年。网址：https://arxiv.org/abs/2203.11171[10] D. Zhou、N. Schärli、L. Hou、J. Wei、N. Scales、X. Wang、D. Schuurmans、C. Cui、O. Bousquet、Q. Le、E. Chi，《由少到多提示使大语言模型能够进行复杂推理》，载于：第十一届国际学习表征会议（ICLR），2023 年。网址：https://arxiv.org/abs/2205.10625[11]S. Yao、D. Yu、J. Zhao、I. Shafran、T. L. Griffiths、Y. Cao、K. Narasimhan，《思维树：利用大语言模型进行审慎的问题求解》，载于：神经信息处理系统进展，第 36 卷，2023 年。arXiv:2305.10601。网址：https://arxiv.org/abs/2305.10601[12] A. Prasad、H. Kokel、A. Medhekar、M. Sap、M. Bansal、K. R. Chandu，《ADaPT：按需使用语言模型进行分解与规划》，载于：计算语言学协会北美分会会议论文集，2024 年，第 4226–4252 页。网址：https://aclanthology.org/2024.findings-naacl.264/[13] J. Chen、Y. Lin、Z. Zhao、H. Liu、X. Bai、Q. Liu、R. Zhang，《FunCoder：释放函数在代码生成中的强大能力》，载于：第 62 届计算语言学协会年会论文集，2024 年。网址：https://openreview.net/forum?id=cFqAANINgW[14] K. U. Qasim、J. Zhang、H. S. U. Rehman，《与问题复杂度无关的思维递归分解；不受问题复杂度限制的递归分解框架》（2025）。arXiv:2601.04210。网址：https://arxiv.org/abs/2601.04210[15] K. U. Qasim、J. Zhang、T. Alsahfi、A. U. R. Butt，《逻辑思维的递归分解：实现卓越推理的框架及大语言模型中的知识传播，采用知识传播进行递归任务分解的 rDoLT 框架（2025）。arXiv:2501.02026。URL https://arxiv.org/abs/2501.02026[16] M. Geng, S. Pateria, B. Subagdja, A.-H. Tan，HiSOMA：一种分层多智能体模型，将自组织神经网络与多智能体深度强化学习相结合，《专家系统及其应用》252 (2024) 124117，采用时序与结构分解策略进行分层任务分解。doi:10.1016/ j.eswa.2024.124117。[17] E. Zelikman, Y. Wu, J. Mu, N. D. Goodman，STaR：通过推理来引导推理，《神经信息处理系统进展》35 (2022)。arXiv:2203.14465。[18] A. Singh, J. D. Co-Reyes, R. Agarwal, A. Anand, P. Patil, X. Gar- cia, P. J. Liu, J. Harrison, J. Lee, K. Xu, A. Parisi, A. Kumar, A. Alemi, A. Rizkowsky, A. Nova, B. Adler, B. Bohnet, G. Elsayed, H. Sedghi, I. Mordatch, I. Simpson, I. Gur, J. Sohl-Dickstein, N. Car- lini, N. Sinha, J. W. Lee, L. Pinto, M. Tao, O. Nachum, P. He, D. Schuurmans, O. Firat, M. Chen，《超越人类数据：扩展语言模型的问题求解自训练》，载于：《机器学习研究汇刊》，2024。arXiv:2312.06585。[19] A. Hosseini, X. Yuan, N. Malber, D. Schuurmans, R. Agarwal, A. Sor- doni，V-STaR：为自教式推理器训练验证器（2024）。arXiv:2402.06457。URL https://arxiv.org/abs/2402.06457[20] D. Zhang, S. Zeng, H. Li, S. Lai, S. Zeng, X. Zhong, J. Li, K. Gao, K. Wei, Z. Lin, ReST-MCTS*：通过过程奖励引导的树搜索进行大语言模型自训练，《神经信息处理系统进展》37 (2024)。URL https://arxiv.org/abs/2406.03816[21] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray 等，《通过人类反馈训练语言模型遵循指令》，载于：《神经信息处理系统进展》第 35 卷，2022 年，第 27730–27744 页。[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov，近端策略优化算法（2017）。arXiv:1707.06347。URL https://arxiv.org/abs/1707.06347[23] Z. Huang, Q. Liu, F. Zhu, L. Zhang, L. Wu，面向稀疏奖励连续空间的无限选项调度分层强化学习，《专家系统及其应用》237 (2024) 121467，采用选项调度的分层强化学习，用于处理稀疏奖励信号。doi:10.1016/j.eswa.2023.121467。[24] A. Kazemnejad, M. Aghajohari, S. Kale, K. Dvijotham, M. M. Khapra, A. Courville, E. Bengio, P. Goel, S. Pitis，VinePPO：通过改进信用分配释放强化学习在大语言模型推理中的潜力，载于：《神经信息处理系统进展》第 37 卷，2024。URL https://arxiv.org/abs/2410.01679[25] X. Wen, Y. Gao, C. Cheng, Y. Yu，具有可验证奖励的强化学习会隐式激励大语言模型中的推理（2025）。arXiv:2506.14245。URL https://arxiv.org/abs/2506.14245[26] ixiao Zhang, C. Zuo, GRPO-LEAD：一种面向难度的强化学习方法，用于提升语言模型中简洁的数学推理（2025）。arXiv:2504.09696[27] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, D. Guo，Deepseekmath：突破开放语言模型数学推理能力的极限（2024）。arXiv: 2402.03300。URL https://arxiv.org/abs/2402.03300[28] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, K. Cobbe，逐步验证，arXiv 预印本 arXiv:2305.20050 (2023)。[29] P. Wang, L. Li, Z. Shao, R. Xu, D. Dai, Y. Li, D. Chen, Y. Wu, Z. Sui，Math-shepherd：无需人工标注，逐步验证并强化大语言模型，载于：第 62 届年度会议论文集

计算语言学协会，2024 年，第 9426–9439 页。URL https://aclanthology.org/2024.acl-long.510/[30] J. Uesato, N. Kushman, R. Kumar, F. Song, N. Siegel, L. Wang, A. Creswell, G. Irving, I. Higgins，通过基于过程和结果的反馈求解数学文字题，arXiv 预印本 arXiv:2211.14275 (2022)。[31] Y. Bengio, J. Louradour, R. Collobert, J. Weston，课程学习，载于：第 26 届国际机器学习会议论文集，2009 年，第 41–48 页。[32] S. Parashar, S. Gui, X. Li, H. Ling, S. Vemuri, B. Olson, E. Li, Y. Zhang, J. Caverlee, D. Kalathil 等，通过从易到难的课程强化学习提升大语言模型推理能力，载于：arXiv 预印本，2025 年。URL https://www.researchgate.net/publication/392531115[33] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, Y. Wu, B. Neyshabur, G. Gur-Ari, V. Misra，使用语言模型求解定量推理问题（2022）。arXiv:2206.14858。URL https://arxiv.org/abs/2206.14858[34] Z. Azerbayev, H. Schoelkopf, K. Paster, M. D. Santos, S. McAleer, A. Q. Jiang, J. Deng, S. Biderman, S. Welleck，Llemma：面向数学的开放语言模型 （2024）。arXiv:2310.10631。URL https://arxiv.org/abs/2310.10631[35] K. Wang, H. Ren, A. Zhou, Z. Lu, S. Luo, W. Shi, R. Zhang, L. Song, M. Zhan, H. Li，MathCoder：在大语言模型中无缝集成代码，以增强数学推理能力 （2023）。arXiv:2310.03731。URL https://arxiv.org/abs/2310.03731[36] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K.Moore, S. Singh 等，SymPy：用于 Python 的符号计算，《PeerJ 计算机科学》3 (2017) e103。