方法链：在不进行训练的情况下扩展测试时间计算

Cong Liu, Jie Wu†, Weigang Wu, Xu Chen, Liang Lin, Wei-Shi Zheng
Sun Yat-sen University, †Temple University
liucong3@mail.sysu.edu.cn, jiewu@temple.edu, wuweig@mail.sysu.edu.cn,
chenxu35@mail.sysu.edu.cn, linliang@ieee.org, wszheng@ieee.org

Abstract

大型语言模型（LLMs）在处理复杂推理任务时往往表现不佳，原因在于它们的训练数据中缺乏深入的洞察力，而这些洞察力通常在公开可获得的文档中缺失。本文介绍了一个创新且直观的提示框架，称为方法链（CoM），通过整合人类的方法学洞察来增强结构化思维，使 LLMs 能够运用扩展推理解决复杂任务。CoM 利用了高级 LLMs 的元认知能力，通过用户定义的方法激活系统性推理，而无需显式微调。实验表明，CoM 超越了具有竞争力的基线，证明了无需训练的提示方法作为复杂推理任务的稳健解决方案的潜力，并通过类似人类的方法学洞察缩小了与人类水平推理的差距。

1 介绍

最近，OpenAI 的 o1 (OpenAI, 2024) 展示了利用一长串思维来提高大型语言模型（LLMs）推理能力的可能性。在这些长时间的思考过程中，OpenAI 的 o1 显示出高水平的认知能力，例如问题分解、错误识别和纠正，这些能力不断引导思维朝正确方向前进。OpenAI 通过强化学习赋予 o1 这些能力。

本文探讨了大型语言模型（LLMs）是否可以仅通过提示，在无需微调指令的情况下，实现跨领域的长篇、结构化推理的类似自我指导能力。这是一个具有挑战性的问题：尽管通过大型数据集进行微调可以广泛提升对指令的遵循能力，但传统的提示通常仅限于特定任务的少样例示例，这受到如上下文长度和信息提取准确性等限制的影响。结果是，尽管纯提示方法具有低成本、快速部署、高样本效率以及避免灾难性遗忘或数据偏见的优势，却很少用于跨领域任务。

我们的工作受到先前关于大型语言模型（LLMs）中元认知知识研究的启发，这指的是对自己推理过程进行推理的能力。教学研究表明，加强元认知知识可以改善人类的推理能力，而在 LLMs 中，通过鼓励自省和自我反思的提示也观察到了类似的益处 (Wang and Zhao, 2024)。此外，微软的 Phi-3 (et al, 2024) 使用了类似“不要幻想”的系统提示来减少幻想，而当 LLMs 识别出所需技能以检索相关示例时，(Didolkar et al., 2024) 表明数学推理得到了改善。这些发现为我们的方法提供了直观和经验上的支持。

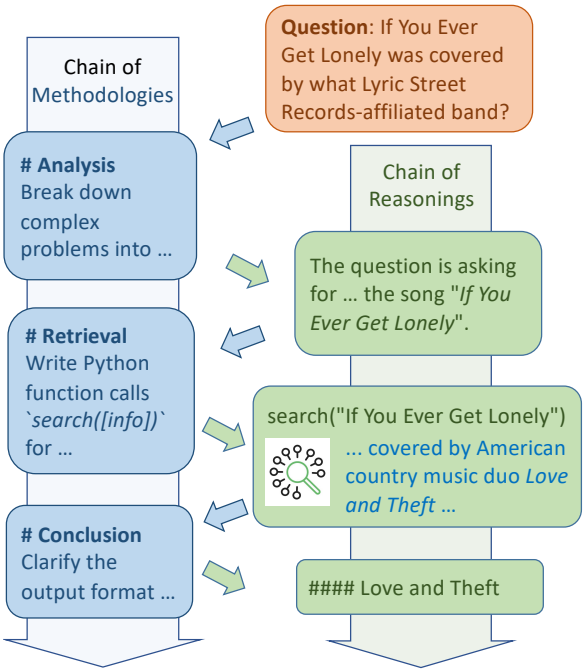

Figure 1: 我们的方法链推理过程的一个示例，其中方法的生成和推理交错进行。一个方法（蓝色）是基于历史推理状态选择的，而下一个推理步骤（绿色）则由先前选择的方法指导。

我们提出了方法链（CoM），这是一种直观的任务无关提示技术，旨在实现跨领域的自我引导推理，而无需进行指令微调。CoM 使用方法论作为催化剂，以刺激 LLM 根据推理历史生成下一个推理步骤。虽然由于训练数据中问题与其相应解决方案之间缺乏深入的见解，LLM 往往在处理复杂推理任务时显得力不从心，但 CoM 通过在每个解决步骤之前插入方法论分析来弥合这一差距，并实现问题到其解决方案的平稳过渡。这利用了 LLM 的元认知

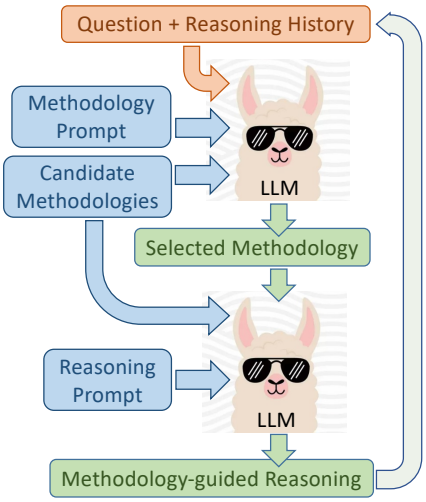

Figure 2: 我们 CoM 框架中的组件及其在方法论推理迭代中的交互。

知识来选择或生成能够证明或解释下一步骤的方法论。

CoM 具有两个关键组件：(1) 一个按照我们的“何时-什么”格式编排的方法列表，该列表便于根据推理历史进行选择，并将其连接到下一个推理步骤；(2) 一个方法-推理循环，迭代地选择下一个方法以在扩展且结构良好的推理路径上指导推理。CoM 的交错方法选择和推理路径的示例如图 1 所示。两个用户定义方法的示例列在图 3 中。

我们的贡献包括简单的 CoM 框架和广泛的实验。CoM 生成结构化的、可解释的和忠实的推理路径。它也具有很高的可扩展性，用户可以通过修改纯文本中的方法列表来增强该框架。我们在两类具有代表性和挑战性的任务上评估了我们的与任务无关的 CoM 框架：数学推理和检索增强生成。实验表明，在多种大型语言模型上，CoM 在这些任务上表现优于竞争基线。

2 方法链

2.1 概述

我们旨在使用提示来激发现有大语言模型 (LLM) 中的高阶认知 (元认知) 知识，使其具备与 OpenAI 的 o1 相同的跨领域自我引导能力，从而成功地在各个领域中执行扩展和结构化的推理序列。这些提示应是不依赖于具体任务的，并有效地引导思维过程。我们发现与方法论相关的提示是实现此目的的理想选择。方法论是任何需要结构化方法来理解、解决问题或进行研究的学科或领域的关键组成部分。它提供了一个框架，确保任务能够一致且有效地执行。

我们的方法链 (CoM) 框架由用户定义的方法列表和方法推理迭代组成。每种方法根据推理历史为下一个推理步骤提供指导。CoM 的推理过程在方法选择步骤和方法指导的推理步骤之间交替进行，如图 1 所示。

方法论列表：与在定义规则和封闭行动空间内操作的 AlphaGo 不同，人类在开放行动空间内的一般问题解决能力更复杂且更具挑战性以进行优化。事实上，人类方法论的积累和演变依赖于基本的过程，如试验和错误、反思以及基于跨不同时代和文明的问题解决经验的自我纠正。为了驾驭这种复杂性，我们通过既定的方法论整合与任务完成相关的人类知识和经验。令 $M = \{m^{(1)}, m^{(2)}, \cdots, m^{(n)}\}$ 表示 n 用户定义的方法论列表。

推理迭代：CoM 为每个问题 Q 最多进行 K 步。在步骤 k 中，当 $1 \leq k \leq K$ 时，我们首先使用提示模板 P_p 提示 LLM_m，以根据推理历史 h_k 选择一种方法 $m_k \in M$：

$$m_k = LLM_p(M, Q, h_k, P_p) \quad (1)$$

，然后使用提示模板 P_r 提示 LLM_r，根据方法论 m_k 和历史 h_k 生成下一个推理序列 r_k：

$$r_k = LLM_r(M, Q, h_k, m_k, P_r) \quad (2)$$

，其中推理历史包含所有先前的推理序列 $h_k = [r_1, r_2, \cdots, r_{k-1}]$。在本文中，我们简单地对 LLM_m 和 LLM_r 都使用相同的指令微调过的大语言模型，在我们框架的应用中保持冻结状态。

2.2 方法论定义

我们的重点在于利用用户定义的方法列表的框架，而不是研究找到一套普遍适用的方法这一哲学问题，其存在性在普遍主义和情境主义之间是一个有争议的话题。从实用的角度看，我们专注于以一种便于方法选择和基于方法推理的方式来表示每种方法。

为了澄清方法和方法论之间的区别，方法指的是实现任务的具体技术或系统程序，而方法论包括指导方法选择和应用的原则和理由。在我们用户定义的列表中，每种方法论都指定两个关键字段：何时和什么。何时字段定义了方法论在推理生命周期中适用的阶段，以及影响方法论选择的背景和因素。什么字段概述了方法论的系统方法、行动选择标准和预期结果。

具体来说，一种方法被定义为 Markdown 格式，包含三个字段：(1) 其名称，(2) 时间：适用的情况和时机，以及 (3) 内容：其规格和细节，包括原则、工具、技术和程序。图 3 提供了两个方法定义的示例。

Analysis
- When: In step 1.
- What: Analyze the category and solution type of the question, list the key facts, variables, relations, constraints with their associated values, and clarify the required output format. Break down complex problems into simpler steps while maintaining critical context. Propose a sequence of methodologies necessary to tackle the remaining reasoning steps iteratively and explain how they are related to the final result.

Retrieval
- When: Fact-based information from the internet is needed.
- What: Write 1-3 line(s) of Python function call(s) `search([information],topk=3)` for each information needed to retrieve. The function `search` has been defined and imported for you, which returns a text summary for the argument `information`. Place your code in a single ```python\n...``` code-block. Finally, accurately simulated the retrieved output by yourself.

Figure 3: 我们在 when-what 格式中的两个示例方法定义。

接下来，我们讨论不同类型的方法论。我们将方法论定义分为三大类型：分析、编码和反思。分析方法论指导 LLM 在组织信息时，如从问题中提取事实、变量、关系、约束和目标；将初始问题分解为可管理的子问题；规划行动的顺序；以及总结、重组和提炼迄今为止获得的信息。编码方法论促使 LLM 生成用于求解器执行的形式化语言以获得准确结果，或通过调用附加于求解器的预定义函数来使用外部工具（例如，搜索引擎）。反思方法论鼓励 LLM 通过自我反思或自我验证识别错误并提供建设性反馈，从而调整方法并为后续步骤提出替代策略。附录中的图 8 列出了我们在实验中使用的任务无关的方法论定义。

总之，方法论的使用具有多重目的：（1）提供人为输入的方法论以激发 LLM 的元认知能力，弥补其训练数据在复杂推理任务中缺乏深入见解的不足；（2）通过解释或论证在当前推理情境与下一步的解决方案之间建立自然连接；以及（3）为下一步提供有见地的猜测，避免诸如 MCTS (Qi et al., 2024) 和 RL (OpenAI, 2024; Snell et al., 2024; Zelikman et al., 2022) 等随机搜索方法的极其复杂性，这些方法在一个比 AlphaGo 等游戏中大得多的一般推理空间中运行。

最后，我们的框架设计旨在易于扩展：用户可以通过更新纯文本中的方法定义列表，使其在一般思维方面更加全面，或根据特定技能组对框架进行定制，以准确地针对特定任务。

2.3 方法论-推理迭代

如图 2 所示，CoM 在最大 K 次迭代中交替提示 LLM 生成下一个方法论以及基于下一个方法论的推理序列。

第一个提示指示 LLM 选择下一步推理步骤的方法论。这个提示将用户定义的方法论定义列表、问题、以往基于方法论的推理序列的历史以及提供有关推理阶段和输出格式的说明连接起来，使得 LLM 能够为任务选择最合适的方法论。

第二个提示包括第一个提示中的所有信息，以及使用第一个提示选择的方法。它指示 LLM 在推理时遵循选定方法中概述的指导。此外，第二个提示要求输出包括以下元素：(1) 通过重述其名称来确认所选方法，(2) 实施该方法的思维链推理过程或程序，以及 (3) 推理的总结结果或程序的猜测输出。

在第二个提示之后，调用求解器来对 LLM 的输出进行后处理。这个步骤促进了 LLM 的编程能力 (Chen et al., 2023) 。目前，我们仅实现了一个 Python 解释器，当在输出中检测到 Python 代码块时，它会被触发。这个解释器在一个安全的环境中执行代码，该环境预先导入了几个常用软件包。执行后，LLM 响应中程序的预测输出会被代码执行的实际标准输出替换。这种方法确保了在需要计算的任务上能够进行准确的推理，比如数学任务，有效地实现了人类的方法论："涉及复杂计算的任务，应使用计算器。" 此外，它允许在推理过程中通过 Python API 进行各种类型的工具使用，包括网络搜索、知识库检索，甚至是调用其他 LLM 或操作 LLM 自身的推理过程 (Cao et al., 2023) 。

我们的 Python 解释器在一个沙盒环境中执行代码，该环境作为一个具有安全全局范围的新进程运行。在这个环境中，代码只能访问有限的一组内置函数，并从预定义的包列表中导入。我们为每个进程强制设置 1 分钟的超时时间，因为我们通过实验证明，较大的超时时间并不能显著提高我们实验任务的性能。用户可以通过在 Python 解释器中添加相应的方法定义和实现相关函数来扩展 CoM 框架的工具使

用能力。例如，为了启用谷歌搜索，可以添加一个方法定义，指定一个名为“search”的函数的存在及其参数的意义，然后在 Python 解释器的全局范围内实现并添加该函数。

我们用于方法选择和基于方法的推理的提示在附录 A.1 的图 7 中提供。

3 相关工作

提示设计大量研究探讨了各种提示设计以增强大型语言模型的推理能力。值得注意的方法包括思维链 (Wei et al., 2022) 、从简单到复杂 (Zhou et al., 2023) 、自我一致性 (Wang et al., 2023b) 、以及思维树 (Cao et al., 2023) 。提高问题特定性能的方法包括问题重新措辞、任务划分、验证、符号植根 (Lyu et al., 2023; Xu et al., 2024a; Wang et al., 2023a; Zelikman et al., 2022; Wang et al., 2024) 、推理链的真实性和忠实性验证 (Wang et al., 2024) ，以及明确分离知识检索和推理步骤以组织决策过程 (Jin et al., 2024) 。

迭代提示以往的研究也探讨了利用迭代提示方法来构建推理过程。例子包括 Self-Refine (Madaan et al., 2023) 、IRCoT (Trivedi et al., 2023) 、iCAP (Wang et al., 2022) 、MetaGPT (Hong et al., 2024) 和 Chain of Ideas (Anonymous, 2024b) 。这些方法通常依赖于预定义的、硬编码的操作来引导推理。相比之下，我们的工作引入了一个任务无关的框架，利用大语言模型的元认知能力，基于推理历史动态选择方法。此外，尽管之前的工作注重生成下一个推理步骤，我们的方法采用一种先理后行的风格，其中模型在执行特定方法之前，会自省性地论证为何需要这种方法。这反映了人类的元认知过程，且将我们的工作与隐式背景感知的令牌生成区分开来。

基于元认知的几个当代作品与我们的方法密切相关。思想缓冲区 (Yang et al., 2024c) 从先前完成的任务中提取高级指导方针，并将其存储在缓冲区中以供将来重用，通过将第二级慢速思维简化为第一级快速思维，实现从经验中学习并提高效率。然而，与我们的工作不同，其高级指导方针包含特定问题的推理链或代码模板，这些是为特定任务（如复杂的多查询任务）量身定制的。基于技能的 CoT (Didolkar et al., 2024) 通过用相应的技能标记问题、将其聚类以减少冗余、并在推理期间检索技能相关的示例进行上下文学习，探索 LLMs 在数学问题解决中的元认知能力。归纳增强生成 (Zhang et al., 2023b) 识别问题中的关键概念，并使用归纳提示模板提取其相关概念和共同属性，从而促进更准确的推理过程。

Table 1: 我们实验中使用的 LLM。最后三个 LLM 的实验结果在附录 A.2 中报告。

LLM	Size
DeepSeek-V3 (DeepSeek-AI, 2024)	671B
Qwen2-72B-Instruct (Yang et al., 2024a)	72B
Qwen2.5-7B-Instruct (Yang et al., 2024a)	7B
Macro-o1 (Zhao et al., 2024)	7B
Yi-1.5-9B-Chat (01.AI: Alex Young, 2024)	9B
InternLM2.5-7B-chat (Cai et al., 2024)	7B
GLM-4-9b-chat (GLM, 2024)	9B

基于搜索的 rStar (Qi et al., 2024) 引入了一种自我博弈互推理方法，显著提高了小型语言模型的推理能力而无需微调。该方法采用昂贵的蒙特卡罗树搜索（MCTS）结合一组五个推理诱导提示。

基于训练的方法最后，已经开发出基于训练的方法来使大语言模型（LLMs）能够处理长链的思维。例如，STaR (Zelikman et al., 2022) 通过对推理历史的迭代训练，导向正确答案，以帮助模型解决日益复杂的问题。类似地，(Snell et al., 2024) 通过使用包含束搜索、前瞻搜索和最佳 N 认证器的强化学习对小模型进行微调，以执行更多的推理步骤。ReST-MCTS (Zhang et al., 2024) 将过程奖励指导与树搜索 MCTS 整合，以收集更高质量的推理轨迹，而 AFlow (Anonymous, 2024a) 则通过迭代地改进特定任务的工作流程。这些方法突出了基于训练的方法的潜力，但通常需要大量的计算资源。

4 实验

4.1 实验设置

我们评估了 CoM 中两个部分的有效性：方法选择和方法指导推理。

大型语言模型如表 1 所列，我们报告了在相对较大和较小的大型语言模型上进行的实验结果，以及最近一个类似于 OpenAI 的 o1 的开源模型，名为 Macro-o1，这个模型是在过滤后的 Open-O1 CoT 数据集 (Team, 2024) 、Macro-o1 CoT 数据集和 Macro-o1 指令数据集的组合上微调的 Qwen2-7B-Instruct。我们使用 Siliconflow (sil) 和百度云 (clo) 提供的大型语言模型 API，设置为：max_tokens=1024, temperature=0.2, top_k=40, top_p=0.95, n=1。

数据集我们在表格 2 列出的数据集的测试集上，使用相同的方法定义（图 8 ）来评估 CoM。

AIME：美国数学邀请赛考试的 1983-2024 部分，包含复杂的代数方程、几何难题和高级数论问题，用于评估数学理解和解决问题的能力。

GSM8K：语言多样的小学数学文字题，需要 2 到 8 步基本计算来解决。

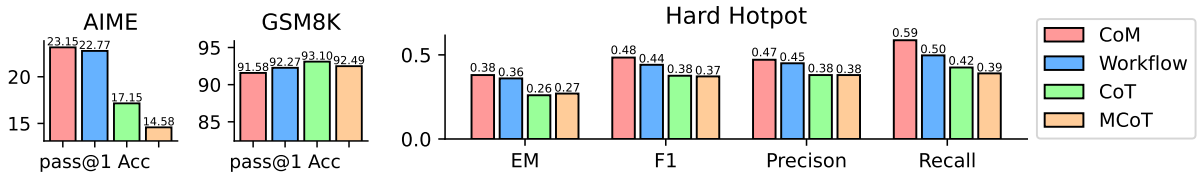

Figure 4: Qwen2-72B-Instruct 在 AIME、GSM8K 和 Hard HotpotQA 上的结果。

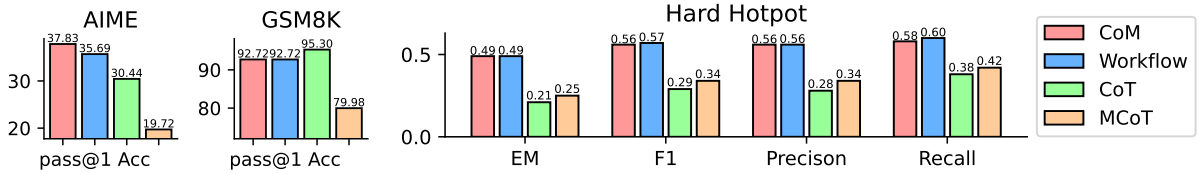

Figure 5: DeepSeek-V3 在 AIME、GSM8K 和 Hard HotpotQA 上的结果。

Table 2: 我们实验中使用的数据集。

Dataset	Size
AIME (Zhang et al., 2023a)	933
GSM8K (Cobbe et al., 2021)	1319
MATH-500 (Lightman et al., 2023)	500
ARC (Clark et al., 2018)	1172
HotpotQA (GLM, 2024)	100

MATH-500：500 个问题来自 OpenAI 创建的 MATH 基准测试。

HotpotQA：一个包含多跳、多步骤问答的数据集的难点部分。我们通过仅向大型语言模型（LLMs）展示问题来模拟增强型检索生成 (Anonymous, 2025) 。当 LLMs 生成调用带有关键词的搜索代码时，我们使用模糊字符串匹配来检索最相似的前 k 个支持性事实。

ARC：AI2 推理挑战数据集，这是一种多项选择问答数据集，包含来自 3 到 9 年级科学考试的问题。

4.2 基线

我们使用零样本提示来评估 CoM，因为少样本方法依赖于特定任务的例子，这使得它们不适合跨领域比较。我们将 CoM 与三个基准进行比较，这些基准使用最近的提示技术和相同的方法定义（图 8 ），以及 Macro-o1 (Zhao et al., 2024) ，一个类似于 OpenAI o1 的最新开源模型。

CoT (Wei et al., 2022) 提示 LLM 生成推理步骤链，并显示最终结果格式。

MCoT 提供了与 CoM 相同的方法定义，并配以 CoT 指令指导 LLM 以适当的顺序使用这些方法。MCoT 评估方法在单轮、非互动环境中是否能够增强推理能力，借鉴了从最少到最多 (Zhou et al., 2023) 和元认知提示 (Wang and Zhao, 2024) 的理念。

无论是 CoT 还是 MCoT 都只是提示一次大型语言模型，并不允许代码生成，因为需要第二次提示来合成代码输出。

流程类似于 CoM，但每个任务使用固定的方法序列，该序列来自 CoM 选择的最常见序列（表 4 ）。它引导 LLM 通过多个推理回合，使用 AIME、GSM8K 和 MATH 的 [分析、编码、变异、结论] 序列，和 Hard Hotpot 及 ARC 的 [分析、检索、结论] 序列。流程结合了 Program-of-Thoughts (Chen et al., 2023) 、Cognitive Prompting (Wang and Zhao, 2024) 、工作流/管道 (Jin et al., 2024; Anonymous, 2024b) 和 RAG (Anonymous, 2025) 的理念。

4.3 性能比较

使用大型语言模型 Qwen2-72B-Instruct，图 4 显示 CoM 在 AIME 和 Hard Hotpot 上表现优于基准方法，其准确率和 F1 分别比 CoT 提高 38.5 % 和 28.7 % 。图 5 的 DeepSeek-V3 也显示了类似的结果。然而，在 GSM8K 上，CoT 略优于 CoM，这可能是由于其简单性和基准泄漏 (Xu et al., 2024b) 。针对任务进行优化的 Workflow 排名第二，而 MCoT 的结果表明单次提示方法的益处最小。

CoM 的表现对 LLM 的元认知能力（即动态选择方法的能力）较为敏感。虽然 Workflow 通过任务特定提示获益（其子任务与 CoM 所选择的最常成功序列相一致，如表 4 所示），但 CoM 仍然是任务无关的，因此更加具有泛化能力。重要的是，在复杂任务上（例如，使用 Qwen2-72B-Instruct 和 DeepSeek-V3 的 AIME）以及在 5 个任务中使用较小的 Qwen2.5-7B-Instruct 的 4 个任务上，CoM 的表现优于 Workflow（图 6 ）。这表明，随着任务复杂性或模型能力的增长，动态选择的优势愈发明显。

如图 4 所示，与任务专门优化的工作流相比，CoM 在 AIME 上的准确性高 1.7 % ，在 Hard Hotpot 上高 9.8 % ，展现了 CoM 在方法选择

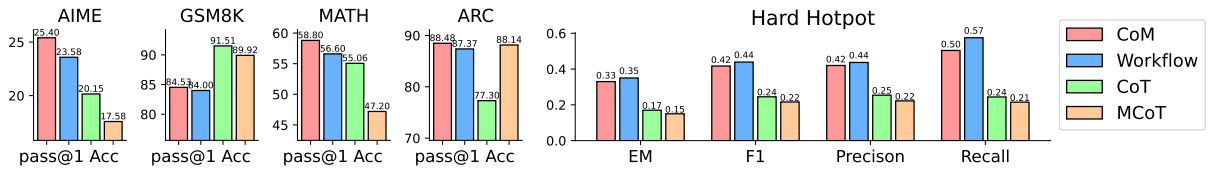

Figure 6: 小型 LLM（如 Qwen2.5-7B-Instruct）在 AIME、GSM8K、MATH、ARC 和 Hard HotpotQA 上的结果。

Table 3: Macro-o1 和 Qwen2.5-7B-Instruct 在 AIME 任务和困难 Hotpot 上的表现

	AIME	Hard Hotpot			
	Acc	EM	F1	Prec	Rec
Macro-o1					
CoT	14.47	0.12	0.20	0.20	0.27
MCoT	10.50	0.09	0.19	0.19	0.25
Qwen2.5-7B-Instruct					
CoT	20.15	0.17	0.25	0.25	0.24
MCoT	17.58	0.15	0.22	0.22	0.22
CoM	25.4	0.33	0.42	0.42	0.51

Table 4: AIME 上排名前 52.2 的% 精选方法序列

	Methodology Sequence
22.0 %	Analysis Coding Validation Conclusion
16.2 %	Analysis Coding Conclusion
5.4 %	Analysis Coding Validation Reflection Flexibility Conclusion
4.4 %	Analysis Coding Validation Reflection Conclusion
4.3 %	Analysis Coding Validation Reflection Flexibility Validation Conclusion

上的优越灵活性。这突出显示了 LLMs 在选择合适的方法序列中的元认知能力的有效性，并验证了我们逐步推理方法的可行性。

对于较小的 LLM Qwen2.5-7B-Instruct（图 6），CoM 在 AIME、MATH 和 ARC 上仍然表现最佳。可能由于基准数据泄露 (Xu et al., 2024b)，CoM 和 Workflow 的准确性都较低。在 Hard Hotpot 上，CoM 略逊于 Workflow，这表明较小的模型在选择方法上具有较弱的元认知能力。

CoM 的有效性取决于大型语言模型的元认知能力。如我们的附录图 9 到 11 所示，缺乏这些能力的模型在方法选择上存在困难。然而，图 6 显示，Qwen2.5-7B-Instruct（7B 参数）在多个任务上始终优于基准。这表明元认知能力可能在较小规模时就会出现。

最后，我们在表 3 中比较了 CoM 与 Macro-o1 (Zhao et al., 2024)。结果表明，在 AIME 和 Hard Hotpot 上微调未能提高 Macro-o1，表明微调数据的泛化性不足。对于 Macro-o1，我们仅评估了单轮方法（CoT/MCoT），因为它缺乏多轮次指令跟随能力。

Table 5: COM 方法的消融研究结果

CoM	AIME (%)	Hard Hotpot
No Ablation	25.4	0.4174
- Interpreter	14.1 (-44.5 %)	0.25 (-40.2 %)
- Analysis	18.7 (-26.6 %)	0.38 (-8.4 %)
- Coding	23.3 (-8.3 %)	0.38 (-8.8 %)
- Retrieval	-	0.22 (-46.8 %)
- Validation	23.9 (-7.2 %)	0.4 (-3 %)
- Reflection	22.8 (-10.5 %)	0.38 (-9 %)
- Synthesis	23 (-9.3 %)	0.4 (-3 %)

4.4 方法论选择模式

我们分析了实验中的推理历史，以识别被 CoM 选择的最常见的方法学序列。表格 4 展示了前五种模式，这些模式占用了 CoM 的响应中使用 Qwen2-72B-Instruct 的 52 %。

在 22.0 % 的案例中，CoM 遵循结构化的方法：分析、生成和执行代码、验证结果以及得出结论。在 16.2 % 的案例中，该模型跳过了验证，表明对其代码有高度信心。其余的模式涉及额外的错误纠正步骤，表明可能存在验证问题。这些发现表明，LLMs 在解决问题时通过规划其推理步骤展示了元认知能力。

4.5 消融研究

我们研究了我们组件模型（CoM）中每个组成部分的相对重要性，包括 Python 解释器以及我们使用的每种方法。在这里，与排除代码方法导致 CoM 无法生成代码相比，移除 Python 解释器仍然允许 LLM 生成代码，但 LLM 需要自己猜测代码的输出，而不是通过解释器。我们的消融研究使用了 Qwen2.5-7B-Instruct 进行。

如表 5 所示，解释器对于这两项任务都非常重要，这表明在不使用代码解释器的情况下，LLM 猜测出的用于数学计算和知识检索的代码输出是不可靠的。其次，对于 AIME 中的困难数学问题，系统分析问题中的数据和约束对推理的正确性至关重要。对于 AIME，我们提供的所有方法都有效，每种方法在准确性上贡献了 7-10 % 的提高。在 AIME 中，我们出于实验简化考虑禁用了检索。对于 Hard Hotpot，因为推理依赖于检索信息，检索显然是最重要的方法。

Table 6: 实验的平均速度（每次迭代按秒计算，乘以 50）

	AIME	GSM8K	Hard Hotpot
Macro-o1			
CoT	96.0	33.5	19.0
MCoT	84.5	42.5	20.0
Qwen2.5-7B-Instruct			
CoM	91.0	34.0	50.5
Workflow	36.0	18.0	21.5
CoT	19.0	5.0	3.5
MCoT	19.0	8.0	3.5

4.6 误差分析

CoM 中的错误是常规的大模型错误，如幻觉、误解和遵循指令错误。我们手动检查了 GSM8K 数据集中 CoM 的前 10 个错误案例。我们发现，在大多数情况下，方法选择并不完美。三个错误案例是由于幻觉引起的，其中错误答案是直接给出的，没有经过必要的计算过程。两个案例是由于自然语言到数学的翻译错误；例如，“早出生”被翻译为年龄减少。三个案例是由于语言理解错误；例如，“重新开始下载”被理解为“继续下载”，“每秒”被理解为“从第二秒”。在一个错误案例中，初始计算是正确的，但随后一个验证步骤导致错误，因为大模型认为“份量”必须是一个整数。在一个错误案例中，尽管方法定义中包含生成单独代码块的指令，大模型生成了多个代码块。

4.7 效率

我们在速度方面以总推理时间和在成本方面以推理次数来考察推理效率。表格 6 显示，在 AIME 中，CoM 的速度是 CoT 的约 5 倍，而在 Hard Hotpot 中则是 7 倍。然而，就总运行时间而言，CoM 与微调模型可媲美。这是因为总运行时间主要由完成标记主导，而 CoM 的多轮调用（2-16 个短提示）与 Macro-o1 的单次长调用相比，延迟是可比的，因为 CoM 的提示是简洁的。

Com 在性能和效率之间取得了良好的平衡。在性能方面，CoM 相比 CoT 显示出显著的提升（例如，在 AIME 上提升了 25 %，在 MATH 上提升了 7 %，在 ARC 上提升了 14 %）。虽然与 Workflow 相比的改进较小，但 Workflow 是一个高度优化的任务特定基线——这使得 CoM 的竞争性能值得注意。在效率方面，尽管 CoM 每个问题使用了约 10 个提示（表格 7），但每个提示均引发简短的响应（图 7），使得延迟与微调模型相当（表格 6），远低于基于搜索的方法（例如，ToT）。

表格 7 比较了 CoM 与 Workflow 提示的数量。结果显示，虽然我们设置了最大迭代 $K = 8$，CoM 平均上在步数上比最大迭代 2*K 更早停止，为较难的 AIME 问题生成了更多的推理步骤，而对于较简单的 GSM8K 问题则生成了更少的步骤。

Table 7: 每个问题的平均提示次数（不是标记数）

	AIME	GSM8K	Hard Hotpot
CoM	2× 5.76	2× 3.99	2× 5.98
Workflow	4	4	3

4.8 实验总结

关于复杂数学问题（AIME 和 GSM8K）的实验，以及多跳问题回答（HotpotQA）评估了 CoM 在方法选择和引导推理中的有效性，使用了一个 72B，一个 7B LLM，以及一个针对结构性推理微调的 LLM。

结果表明，我们的 CoM 在提高两个具有挑战性的任务的表现上相较于近期的提示工程方法的基准更加有效。该结果支持我们的假设，即我们可以使用一种无需训练的解决方案，将人类的方法论见解整合起来，以增强 LLM 在复杂推理中的表现。

方法选择模式显示，CoM 有效地生成合理的方法序列，引导其推理朝正确的方向发展。错误分析表明，常见的 LLM 问题是 CoM 出现大多数错误的原因。最后，消融研究证实我们采用的方法对于解决复杂的推理任务至关重要。

5 结论与未来工作

本文通过模拟认知控制过程和利用用户定义的方法提高了 LLMs 在复杂任务中的推理能力，使其能够有效地处理复杂的推理任务，而无需大量的再训练。研究结果包括：（1）LLMs 展示了潜在的元认知能力，这些能力可以通过结构化的、以理由为驱动的提示激活，消除了微调的需求；（2）生成明确的方法理由可以改善可追溯性和任务理解，提高零样本准确率和跨领域适应性，而这通常受限于有限的上下文示例。

本文的主要目标是研究使用 LLM 自生成的思维指导方针（例如，方法论）来引导其推理过程的可行性。未来的工作包括微调一个小的适配器以提高元认知能力，并将设计工作从提示框架转向方法论的工程化。

6

局限性

本文提出的方法假设 LLM 具有元认知能力。我们发现，尽管其他 LLM 在各种基准测试中表现出竞争力，但即便经过大量提示调优，这些模型在方法选择上却失败了。例如，其中一个 LLM 始终选择其最初选择的第一种方法。

附录 A.2 中提供了其他实验结果，这些结果说明了这些失败。

目前，我们的方法在每个步骤需要两个不同的提示：一个用于方法选择，另一个用于基于方法的推理。我们尝试将这两个提示整合为一个单一的提示；然而，我们发现，即使是我们测试过的最先进的 LLM，包括 DeepSeek-V3，在处理这种更复杂的组合提示时也遇到了困难。我们预计，随着未来 LLM 在遵循指令能力上的进步，将能够使用单一提示，从而提高我们方法的效率。

我们框架中包含的方法并不详尽，还留有空间供未来研究扩展和完善这个列表。纳入更广泛的策略可以提高 CoM 框架的适应性和稳健性，从而开辟新的探索和改进途径。

7

伦理声明

这项工作完全符合 ACL 伦理政策。据我们所知，本文不存在伦理问题。

References

https://cloud.baidu.com/.

https://siliconflow.cn/.

Chao Li Chengen Huang Ge Zhang Guanwei Zhang Guoyin Wang Heng Li Jiangcheng Zhu Jianqun Chen Jing Chang Kaidong Yu Peng Liu Qiang Liu Shawn Yue Senbin Yang Shiming Yang Wen Xie Wenhao Huang Xiaohui Hu Xiaoyi Ren Xinyao Niu Pengcheng Nie Yanpeng Li Yuchi Xu Yudong Liu Yue Wang Yuxuan Cai Zhenyu Gu Zhiyuan Liu Zonghong Dai 01.AI: Alex Young, Bei Chen. 2024. Yi: Open Foundation Models by 01.AI. Preprint , arXiv:2403.04652.

Anonymous. 2024a. AFlow: Automating agentic workflow generation. In The Thirteenth International Conference on Learning Representations (ICLR) .

Anonymous. 2024b. Chain of ideas: Revolutionizing research in idea development with LLM agents. In The Thirteenth International Conference on Learning Representations (ICLR) .

Anonymous. 2025. Inference scaling for long-context retrieval augmented generation. In The Thirteenth International Conference on Learning Representations (ICLR) .

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaxing Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Yining Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu, Yingtong Xiong, Chao Xu, Ruiliang Xu, Hang Yan, Yirong Yan, Xiaogui Yang, Haochen Ye, Huaiyuan Ying, Jia Yu, Jing Yu, Yuhang Zang, Chuyu Zhang, Li Zhang, Pan Zhang, Peng Zhang, Ruijie Zhang, Shuo Zhang, Songyang Zhang, Wenjian Zhang, Wenwei Zhang, Xingcheng Zhang, Xinyue Zhang, Hui Zhao, Qian Zhao, Xiaomeng Zhao, Fengzhe Zhou, Zaida Zhou, Jingming Zhuo, Yicheng Zou, Xipeng Qiu, Yu Qiao, and Dahua Lin. 2024. Internlm2 technical report. Preprint , arXiv:2403.17297.

Shulin Cao, Jiajie Zhang, Jiaxin Shi, Xin Lv, Zijun Yao, Qi Tian, Lei Hou, and Juanzi Li. 2023. Probabilistic tree-of-thought reasoning for answering knowledge-intensive complex questions. In Findings of the Association for Computational Linguistics: EMNLP 2023 , pages 12541–12560, Singapore. Association for Computational Linguistics.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. Transactions on Machine Learning Research .

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. arXiv:1803.05457v1 .

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168 .

DeepSeek-AI. 2024. Deepseek-v3 technical report. https://arxiv.org/pdf/2412.19437.

Aniket Rajiv Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy P Lillicrap, Danilo Jimenez Rezende, Yoshua Bengio, Michael Curtis Mozer, and Sanjeev Arora. 2024. Metacognitive capabilities of LLMs: An exploration in mathematical problem solving. In AI for Math Workshop @ ICML 2024 .

Marah Abdin et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. Preprint , arXiv:2404.14219.

Team GLM. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. Preprint , arXiv:2406.12793.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In The Twelfth International Conference on Learning Representations .

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. 2024. Disentangling memory and reasoning ability in large language models. Preprint , arXiv:2411.13504.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. ICLR 2024 .

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers) , pages 305–329, Nusa Dua, Bali. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In NeurIPS .

OpenAI. 2024. Learning to Reason with LLMs.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. In Arxiv .

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. Preprint , arXiv:2408.03314.

O. Team. 2024. https://github.com/open-source-o1/open-o1.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) , pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Boshi Wang, Xiang Deng, and Huan Sun. 2022. Iteratively prompt pre-trained language models for chain of thought. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing , pages 2714–2730, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jianing Wang, Qiushi Sun, Xiang Li, and Ming Gao. 2024. Boosting language models reasoning with chain-of-knowledge prompting. In The 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) , pages 4958–4981, Bangkok, Thailand. Association for Computational Linguistics.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) , pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In The Eleventh International Conference on Learning Representations .

Yuqing Wang and Yun Zhao. 2024. Metacognitive prompting improves understanding in large language models. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers) , pages 1914–1926, Mexico City, Mexico. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In Advances in Neural Information Processing Systems .

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024a. Faithful logical reasoning via symbolic chain-of-thought. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) , pages 13326–13365, Bangkok, Thailand. Association for Computational Linguistics.

Ruijie Xu, Zengzhi Wang, Run-Ze Fan, and Pengfei Liu. 2024b. Benchmarking benchmark leakage in large language models. arXiv preprint arXiv:2404.18824 .

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang,

Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024a. Qwen2 technical report. arXiv preprint arXiv:2407.10671 .

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024b. Large language models as optimizers. In The Twelfth International Conference on Learning Representations .

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. 2024c. Buffer of thoughts: Thought-augmented reasoning with large language models. arXiv preprint arXiv:2406.04271 .

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Conference on Empirical Methods in Natural Language Processing (EMNLP) .

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In Advances in Neural Information Processing Systems , volume 35, pages 15476–15488. Curran Associates, Inc.

Dan Zhang, Sining Zhoubian, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Rest-mcts*: Llm self-training via process reward guided tree search. Thirty-eighth Conference on Neural Information Processing Systems (NeurIPS).

Xingyuan Zhang, Philip Becker-Ehmck, Patrick van der Smagt, and Maximilian Karl. 2023a. Action inference by maximising evidence: Zero-shot imitation from observation with world models. In Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS) .

Zhebin Zhang, Xinyu Zhang, Yuanhang Ren, Saijiang Shi, Meng Han, Yongkang Wu, Ruofei Lai, and Zhao Cao. 2023b. IAG: Induction-augmented generation framework for answering reasoning questions. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing , pages 1–14, Singapore. Association for Computational Linguistics.

Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. Marco-o1: Towards open reasoning models for open-ended solutions. Preprint , arXiv:2411.14405.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In The Eleventh International Conference on Learning Representations .

A 附录

A.1 提示

我们的方法列表展示在图 8 中，我们的方法选择和基于方法的推理提示列在图 7 中。

A.2 附加实验结果

本文提出的方法假设大语言模型（LLM）具备元认知能力。本节在图 9 、10 和 11 中展示了额外的实验结果，揭示尽管一些 LLM 在各种基准测试中表现出竞争力，它们在方法选择上仍面临困难，即便经过广泛的提示调优。例如，其中一个 LLM 始终选择最初识别的第一个方法，这突显了其决策过程中的局限性。

我们进行了实验，使用了表格 **??** 中显示的自一致性 (Wang et al., 2023b) (CoT-SC)。请注意，CoT-SC 在 CoT 之上有改进，但 SC 与 CoM 是正交的。

Table 8: 不同方法的性能结果

Method	AIME	GSM8K	MATH
CoT	20.15	91.51	55.06
CoT-SC	27.12	92.19	68.00
CoM	25.40	84.53	58.80

 # List of Existing Methodologies

{example_methodologies}

{historical_reasoning_steps}

 # Question
- {question}

 # Step Instruction
- You are currently on {cur_step} out of a total of a maximum of {max_steps} steps to solve the question. Please select the most suitable methodology from the list, considering the reasoning history and the timing for applying each methodology. Try to diversify your choices to advance the resolution of the question, and avoid redundant or repeated methodologies. Select and output a single methodology succinctly in the following format, without any additional text:

[Methodology name]
- When: [The timing for applying the methodology]
- What: [The characteristics and details of the methodology]

 # List of Existing Methodologies

{example_methodologies}

{historical_reasoning_steps}

 # Question
- {question}

 # Selected Methodology

{selected_methodology}

 # Step Instruction
- You are currently on {cur_step} out of a total of a maximum of {max_steps} steps to solve the question. Please conduct an innovative next reasoning step for the question, using the selected methodology "{selected_methodology_name}" based on the current reasoning history. Please output your reasoning step succinctly in the following format, without any additional text:

- Methodology: [Name of the selected methodology]
- Reasoning/Code: [A structural and systematic reasoning step that uses the selected methodology and improves on your past reasoning steps. Or a Python snippet with such reasoning inside its comments in a ```python\n...``` code-block.]
- Result: [A short result of your reasoning, or an accurate simulated outpout from the code.]

Figure 7: 我们的方法选择提示（左）和基于方法的推理提示（右）。

Analysis
- When: In step 1.
- What: Analyze the category and solution type of the question, list the key facts, variables, relations, constraints with their associated values, and clarify the required output format. Break down complex problems into simpler steps while maintaining critical context. Propose a sequence of methodologies necessary to tackle the remaining reasoning steps iteratively and explain how they are related to the final result.

Retrieval
- When: Fact-based information from the internet is needed.
- What: Write 1-3 line(s) of Python function call(s) `search([information],topk=3)` for each information needed to retrieve. The function `search` has been defined and imported for you, which returns a text summary for the argument `information`. Place your code in a single ```python\n...``` code-block. Finally, accurately simulated the retrieved output by yourself.

Coding
- When: Coding is necessary.
- What: A standalone Python snippet with structural and systematic reasoning in comments, using the **print** function to output the result. Place your code in a single ```python\n...``` code-block. Finally, accurately simulated the output of your Python snippet by yourself without using a computer.

Validation
- When: A temporary or a final the result is resulting from a previous reasoning step.
- What: Identify the result, and analyze its correctness from a different angle. You may write a test-case function that prints True/False to validate the result and then simulate its output.

Reflection
- When: An error is detected or validation fails.
- What: Analyze the reasoning steps to identify errors and provide constructive self-critic or feedback for improvement.

Flexibility
- When: The previous step fails to obtains the expected result or when a reflective or critic feedback is available.
- What: Adjust the approach based on insights gained and propose alternative strategies for the next steps.

Conclusion
- When: A confident final answer is available, or in the last step.
- What: Clarify the output format required by the question. Compile the reasoning process and generate the answer in the required format.

Figure 8: 我们的方法列表。

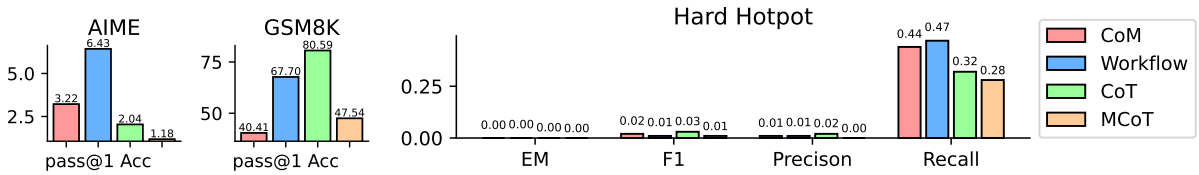

Figure 9: Yi-1.5-9B-Chat (01.AI: Alex Young, 2024) 在 AIME、GSM8K 和 Hard HotpotQA 上的结果。

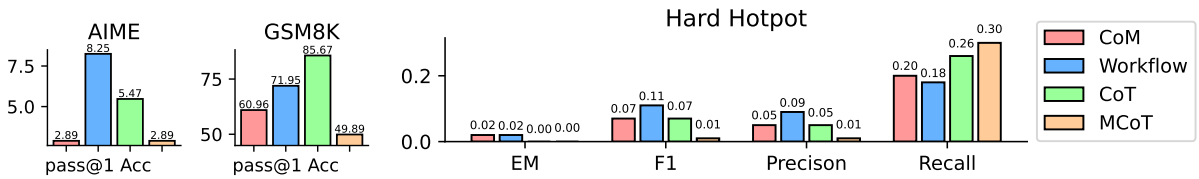

Figure 10: InternLM2.5-7B-chat (Cai et al., 2024) 在 AIME、GSM8K 和 Hard HotpotQA 上的结果。

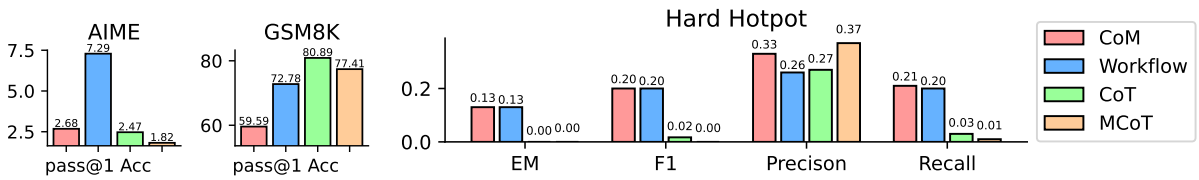

Figure 11: GLM-4-9b-chat (GLM, 2024) 在 AIME、GSM8K 和 Hard HotpotQA 上的结果。