Faithful Chain-of-Thought Reasoning 忠实的思维链推理

Qing Lyu*, Shreya Havaldar*, Adam Stein*, Li Zhang, Delip Rao, Eric Wong, Chris Callison-Burch
University of Pennsylvania
{lyuqing, shreyah, steinad, zharry, exwong, marapi, ccb}@seas.upenn.edu, deliprao@gmail.com

Abstract

While Chain-of-Thought (CoT) prompting boosts Language Models' (LM) performance on a gamut of complex reasoning tasks, the generated reasoning chain does not necessarily reflect how the model arrives at the answer (aka. faithfulness). We propose Faithful CoT, a reasoning framework involving two stages: Translation (Natural Language query → symbolic reasoning chain) and Problem Solving (reasoning chain → answer), using an LM and a deterministic solver respectively. This guarantees that the reasoning chain provides a faithful explanation of the final answer. Aside from interpretability, Faithful CoT also improves empirical performance: it outperforms standard CoT on 9 of 10 benchmarks from 4 diverse domains, with a relative accuracy gain of 6.3% on Math Word Problems (MWP), 3.4% on Planning, 5.5% on Multi-hop Question Answering (QA), and 21.4% on Relational Inference. Furthermore, with GPT-4 and Codex, it sets the new state-of-the-art few-shot performance on 7 datasets (with 95.0+ accuracy on 6 of them), showing a strong synergy between faithfulness and accuracy.

虽然链式思维 (CoT) 提示提升了语言模型 (LM) 在一系列复杂推理任务上的表现，但生成的推理链并不一定反映模型如何得出答案 (即，可信度)。我们提出了可信的链式思维 (Faithful CoT)，这是一个包含两个阶段的推理框架: 翻译 (自然语言查询 → 符号推理链) 和问题解决 (推理链 → 答案)，分别使用语言模型和确定性求解器。这保证了推理链提供了最终答案的可信解释。除了可解释性，可信的链式思维还提高了经验性能: 在来自 4 个不同领域的 10 个基准中，它在 9 个基准上优于标准的链式思维，在数学文字问题 (MWP) 上相对准确度提升了 6.3% ，在规划上提升了 3.4%，在多跳问答 (QA) 上提升了 5.5%，在关系推理上提升了 21.4%。此外，使用 GPT-4 和 Codex，它在 7 个数据集上设定了新的最先进的少量样本性能 (其中 6 个数据集的准确率超过 95.0)，显示出可信度与准确性之间的强大协同作用。[1]

1 Introduction

Complex reasoning tasks, such as commonsense reasoning and math reasoning, have long been the Achilles heel of LMs (Bengio, 2019), until a recent line of work on Chain-of-Thought (CoT) reasoning (Wei et al., 2022; Wang et al., 2022, i.a.) brought striking performance gains. CoT prompts an LM to generate a reasoning chain along with the answer, given only a few in-context exemplars.

复杂推理任务，如常识推理和数学推理，长期以来一直是语言模型的致命弱点 (Bengio, 2019)，直到最近一系列关于链式思维 (CoT) 推理的研究 (Wei et al., 2022; Wang et al., 2022 等) 带来了显著的性能提升。CoT 提示语言模型在仅给出少量上下文示例的情况下生成推理链及其答案。

Besides performance improvement, CoT is also claimed to "provide an interpretable window into the behavior of the model" (Wei et al., 2022). However, it is unclear how accurately these reasoning chains reflect the underlying reasoning process behind the model's prediction, namely, how faithful they are as explanations (Jacovi and Goldberg, 2020, i.a.). In standard CoT, faithfulness is not guaranteed and even systematically violated (Turpin et al., 2023), as the final answer does not necessarily follow from the generated reasoning chain. In other words, CoT can "lie" about the model's true reasoning process. Figure 1 exemplifies such an unfaithful CoT generation from Wei et al. (2022) on GSM8K: the answer "0" is not even mentioned in the reasoning chain. This, along with more examples in Appendix B.1, illustrates that standard CoT does not provide interpretability of how the model predicts the answer.

* Equal contribution.

[1] Our code, data, and prompts are available at https:// github.com/veronica320/Faithful-COT.

除了性能提升，CoT 还被声称为“提供了一个可解释的窗口，观察模型的行为”(Wei et al., 2022)。然而，这些推理链在多大程度上准确反映模型预测背后的推理过程，即它们作为解释的可信度，仍不清楚 (Jacovi and Goldberg, 2020 等)。在标准的链式思维中，可信度并不保证，甚至系统性地被违反 (Turpin et al., 2023)，因为最终答案并不一定遵循生成的推理链。换句话说，链式思维可能会“撒谎”关于模型的真实推理过程。图 1 展示了 Wei et al.(2022) 在 GSM8K 上生成的一个不可信的链式思维示例: 答案“0”甚至没有在推理链中提到。这一点，以及附录 B.1 中的更多示例，说明标准的链式思维并未提供模型如何预测答案的可解释性。

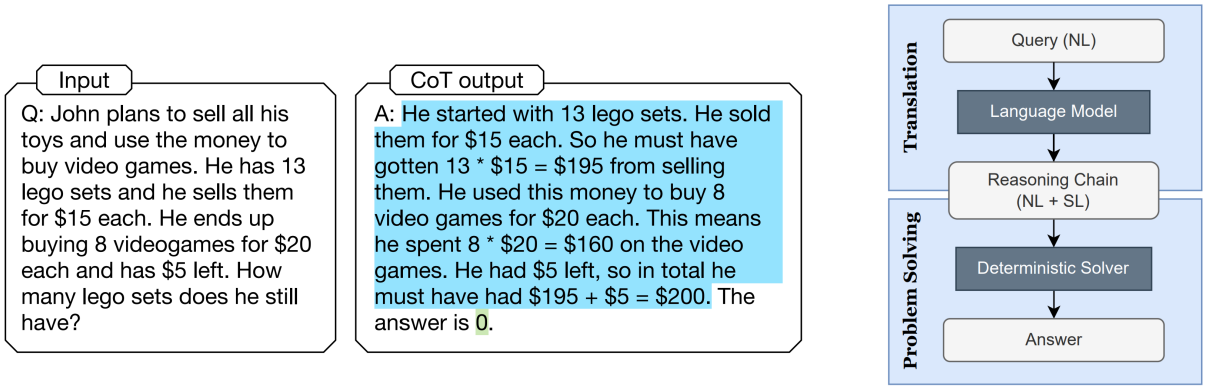

Figure 1: An example of unfaithful output from CoT prompting (Wei et al., 2022) on GSM8K. The answer (green) does not follow from the reasoning chain (blue). 来自链式思维提示 (Wei et al., 2022) 在 GSM8K 上的不可信输出示例。答案 (绿色) 并不遵循推理链 (蓝色)。

Figure 2: An overview of our Faithful CoT framework, consisting of Translation, where an LM translates a query (in NL/Natural Language) into a reasoning chain (which interleaves NL and SL/Symbolic Language), and Problem Solving, where an external solver executes the reasoning chain to derive the answer. 我们的 Faithful CoT 框架的概述，包括翻译，其中一个 LM 将查询 (以自然语言 NL 表示) 翻译为推理链 (交错使用自然语言和符号语言 SL)，以及问题解决，其中一个外部求解器执行推理链以得出答案。

The lack of faithfulness in CoT can be dangerous in high-stake applications because it may mislead people into believing that the model is self-interpretable, while there is no actual causal relationship between the reasoning chain and the answer. Even worse, when an unfaithful explanation looks plausible (i.e., convincing to humans) (Jacovi and Goldberg, 2020), this makes it easier for people (e.g., legal practitioners) to over-trust the model (e.g., a recidivism predictor) even if it has implicit biases (e.g., against racial minorities) (Pruthi et al., 2020; Slack et al., 2020).

CoT 中缺乏忠实性在高风险应用中可能是危险的，因为这可能误导人们相信模型是自我可解释的，而推理链与答案之间实际上并不存在因果关系。更糟糕的是，当一个不忠实的解释看起来似乎合理 (即对人类具有说服力)(Jacovi 和 Goldberg，2020) 时，这使得人们 (例如，法律从业者) 更容易过度信任模型 (例如，累犯预测器)，即使它存在隐性偏见 (例如，针对种族少数群体)(Pruthi 等，2020；Slack 等，2020)。

To address this concern, we propose Faithful CoT , a reasoning framework where the answer is the result of deterministically executing the reasoning chain. Specifically, we break down a complex reasoning task into two stages: Translation and Problem Solving (Figure 2). During Translation, an LM translates a query into a reasoning chain, which interleaves NL and Symbolic Language (SL). The NL component decomposes the original query into multiple simpler, interdependent subproblems. Then, each subproblem is tackled in a task-dependent SL, such as Python, Dat-alog, or Planning Domain Definition Language (PDDL). In the Problem Solving stage, the reasoning chain is executed by a deterministic solver, e.g., a Python/Datalog interpreter, or a PDDL planner, to derive the answer.

为了解决这个问题，我们提出了 Faithful CoT ，这是一个推理框架，其中答案是通过确定性地执行推理链而得出的结果。具体而言，我们将一个复杂的推理任务分解为两个阶段: 翻译和问题解决。将复杂的推理任务分解为两个阶段: 翻译和问题解决 (图 2)。在翻译阶段，一个 LM 将查询翻译为推理链，该推理链交错使用自然语言和符号语言 (SL)。自然语言组件将原始查询分解为多个更简单的相互依赖的子问题。然后，每个子问题在任务相关的符号语言中解决，例如 Python、Datalog 或规划领域定义语言 (PDDL)。在问题解决阶段，推理链由确定性求解器执行，例如 Python/Datalog 解释器或 PDDL 规划器，以得出答案。

Our reasoning chain (outcome of Translation) is guaranteed to provide a faithful explanation of how the final answer is produced (outcome of Problem Solving), therefore making our method more interpretable than standard CoT methods. While interpretability is not the same as correctness (i.e. our method can reveal the reasoning process behind both correct and wrong answers), we find that it does empirically improve correctness: when evaluated on 10 reasoning datasets from 4 diverse domains (MWP, Planning, Multi-hop QA, and Relational Inference), Faithful CoT brings consistent performance gains over three existing baselines, across different LMs and decoding strategies. With Codex, our approach outperforms vanilla CoT on 9 of the 10 datasets, with a relative accuracy gain of 6.3% on MWP, 3.4% on Planning, 5.5% on Multihop QA, and 21.4% on Relational Inference. With GPT-4, our method sets the new SOTA few-shot performance on 7 datasets, with 95.0+ accuracy on 6 of them. This suggests that interpretability does not have to come at the cost of performance; instead, there exists a strong synergy in between.

我们的推理链 (翻译的结果) 保证提供对最终答案生成过程的忠实解释 (问题解决的结果)，因此使我们的方法比标准的 CoT 方法更具可解释性。[2] 虽然可解释性与正确性并不相同 (即我们的方法可以揭示正确和错误答案背后的推理过程)，但我们发现它在经验上确实提高了正确性: 在对来自 4 个不同领域 (MWP、规划、多跳问答和关系推理) 的 10 个推理数据集进行评估时，Faithful CoT 在不同的 LMs 和解码策略下，相比于三个现有基线模型，带来了持续的性能提升。使用 Codex，我们的方法在 10 个数据集中的 9 个上超越了普通的 CoT，在 MWP 上相对准确率提高了 6.3%，在规划上提高了 3.4%，在多跳问答上提高了 5.5%，在关系推理上提高了 21.4%。使用 GPT-4，我们的方法在 7 个数据集上设定了新的 SOTA 少样本性能，其中 6 个数据集的准确率超过 95.0%。这表明可解释性并不一定以牺牲性能为代价；相反，两者之间存在强大的协同作用。

Our key contributions are as follows:

我们的主要贡献如下:

(a) We propose Faithful CoT, a framework that decomposes reasoning into Translation and Problem Solving. The reasoning chain interleaves user-understandable natural language comments and executable symbolic language programs, thus providing faithful interpretability of how the model arrives at the answer.

(a) 我们提出了 Faithful CoT，一个将推理分解为翻译和问题解决的框架。推理链交替使用用户可理解的自然语言注释和可执行的符号语言程序，从而提供模型如何得出答案的忠实可解释性。

(b) Our approach is generalizable to multiple domains beyond arithmetic reasoning and simple symbolic reasoning, thanks to its flexible integration with any choice of SL and external solver. We set the new SOTA performance on 7 out of the 10 reasoning datasets, showing a strong synergy between faithfulness and accuracy.

(b) 我们的方法可以推广到算术推理和简单符号推理之外的多个领域，这得益于它与任何选择的 SL 和外部求解器的灵活集成。我们在 10 个推理数据集中的 7 个上设定了新的 SOTA 性能，显示出忠实性与准确性之间的强大协同作用。

(c) We provide an extensive analysis of the strengths and weaknesses of our method, showing its generalizability across LMs, robustness to the choice of exemplars and prompt phrasing, the pivotal role of the solver, the plausibility of generated reasoning chains, as well as frequent error patterns where it still struggles.

(c) 我们提供了对我们方法的优缺点的广泛分析，展示了其在 LMs 上的可推广性、对示例和提示措辞选择的鲁棒性、求解器的关键作用、生成推理链的合理性，以及它仍然面临的常见错误模式。

[2] Note that we do not claim that the process of generating the reasoning chain itself, i.e., the Translation stage, is interpretable. See more discussion in Limitations. 请注意，我们并不声称生成推理链的过程本身，即翻译阶段，是可解释的。有关更多讨论，请参见局限性。

2 Related Work

Faithfulness. In interpretability, faithfulness (also called fidelity or reliability) means that an explanation should "accurately represent the reasoning process behind the model's prediction", which is a fundamental requirement of an explanation (Harrington et al., 1985; Ribeiro et al., 2016; Gilpin et al.,2018; Jacovi and Goldberg,2020).

It should be contrasted with plausibility (a.k.a. persuasiveness or understandability), which refers to "how convincing an explanation is to humans" (Herman, 2019; Jacovi and Goldberg, 2020). In the context of CoT prompting, a faithful reasoning chain needs to accurately reflect how the model arrives at the final answer, whereas a plausible reasoning chain is one that looks reasonable and coherent to humans. Standard CoT (Wei et al., 2022) generates the reasoning chain in pure NL, which may often look plausible; nevertheless, the final answer does not need to causally follow from the reasoning chain, thus not guaranteeing faithfulness.

忠实性。在可解释性中，忠实性 (也称为保真性或可靠性) 意味着解释应该 "准确地代表模型预测背后的推理过程"，这是解释的基本要求 (Harrington et al., 1985; Ribeiro et al., 2016; Gilpin et al., 2018; Jacovi and Goldberg, 2020)。

Chain-of-Thought-style prompting. In CoT-style prompting, given a complex question Q , an LM is prompted to generate a reasoning chain C along with the final answer A . Specifically, the prompt consists of a few examples of(Q, C, A) triples, called in-context exemplars. This allows pre-trained LMs (e.g., GPT-3 (Brown et al., 2020)) to solve unseen questions with much higher accuracy than standard prompting, where the exemplars do not contain the reasoning chain C .

思维链式提示。在 CoT 风格的提示中，给定一个复杂问题 Q ，语言模型被提示生成一个推理链 C 以及最终答案 A 。具体而言，提示由几个 (Q, C, A) 三元组的示例组成，称为上下文示例。这使得预训练的语言模型 (例如，GPT-3(Brown et al., 2020)) 能够以比标准提示更高的准确性解决未见过的问题，其中示例不包含推理链 C 。

We create a taxonomy of existing CoT-style prompting methods into three types: all-at-once, ensemble-based, and modularized. All-at-once prompting means that the LM produces C and A as one continuous string, without any dependencies or constraints in between. Scratchpad (Nye et al., 2021), standard CoT (Wei et al., 2022), and "Let's think step by step" (Kojima et al., 2022), are all examples of this kind. Ensemble-based prompting is designed to overcome the local optimality issue of the one-shot generation in previous methods by sampling multiple(C, A)pairs and choosing the best answer via strategies like majority voting. Examples include Self-Consistent CoT (Wang et al., 2022), Minerva (Lewkowycz et al., 2022), and DIVERSE (Li et al., 2022), which differ mainly in the voting granularity and the underlying LM. Modularized methods break down Q into subproblems and then conquer them individually (Jung et al., 2022; Qian et al., 2022, i.a.). In particular, Least-to-Most prompting (Zhou et al., 2022) has a similar question decomposition process to ours, but there is still no faithfulness guarantee since the reasoning chain is entirely in NL.

我们将现有的 CoT 风格提示方法创建为三种类型的分类: 一次性、基于集成和模块化。一种一次性提示意味着语言模型将 C 和 A 作为一个连续字符串生成，中间没有任何依赖或约束。Scratchpad(Nye et al., 2021)、标准 CoT(Wei et al., 2022) 和 "让我们一步一步思考" (Kojima et al., 2022) 都是这种类型的示例。基于集成的提示旨在通过采样多个 (C, A) 对并通过多数投票等策略选择最佳答案，克服先前方法中一次性生成的局部最优性问题。示例包括自一致 CoT(Wang et al., 2022)、Minerva(Lewkowycz et al., 2022) 和 DIVERSE(Li et al., 2022)，它们主要在投票粒度和基础语言模型上有所不同。模块化方法将 Q 拆分为子问题，然后逐个解决 (Jung et al., 2022; Qian et al., 2022 等)。特别是，最少到最多提示 (Zhou et al., 2022) 与我们的问答分解过程相似，但由于推理链完全是自然语言，因此仍然没有忠实性保证。

[3] Note that this differs from the notion of faithfulness in the Natural Language Generation (NLG) literature, primarily in what constitutes the ground truth. In interpretability, we talk about the faithfulness of an explanation w.r.t. the model's underlying reasoning mechanism - the ground truth is usually unknown. In NLG, we talk about the faithfulness of the generated text (e.g., a translated sentence, or a summary) w.r.t. some explicit source (e.g., the source sentence, or the full document) - the ground truth is transparent. 请注意，这与自然语言生成 (NLG) 文献中忠实性的概念不同，主要体现在什么构成真实情况。在可解释性中，我们讨论的是解释相对于模型基础推理机制的忠实性——真实情况通常是未知的。在 NLG 中，我们讨论的是生成文本 (例如，翻译句子或摘要) 相对于某个明确来源 (例如，源句子或完整文档) 的忠实性——真实情况是透明的。这应与可信度 (即说服力或可理解性) 相对比，后者指的是 "解释对人类的说服力" (Herman, 2019; Jacovi and Goldberg, 2020)。在 CoT 提示的背景下，忠实的推理链需要准确反映模型如何得出最终答案，而可信的推理链则是对人类看起来合理且连贯的链。标准的 CoT(Wei et al., 2022) 以纯自然语言生成推理链，这通常看起来是可信的；然而，最终答案不需要因果地跟随推理链，因此不保证忠实性。

Concurrent with our work, Chen et al. (2022) and Gao et al. (2022) both generate Python programs (i.e., SL-only reasoning chains) to derive the answer. We want to highlight the following qualitative differences: (a) In terms of motivation, our approach is interpretability-driven, whereas theirs are performance-driven. (b) Our reasoning chain involves a structured decomposition of the problem in NL, allowing users without a programming background to better understand and potentially interact with the system. (c) They only use Python as the SL and only tackle math and simple symbolic reasoning tasks, whereas we demonstrate the generalizability of our approach to multiple symbolic languages and various other domains. In particular, we innovatively recast a diverse set of realistic tasks (Planning, Multi-hop QA, and Relational Inference) into a symbolic representation, which allows us to tackle them with a single framework. (d) We perform a more comprehensive analysis compared to previous work, especially a human evaluation of the reasoning chain correctness.

与我们的工作同时进行，Chen et al.(2022) 和 Gao et al.(2022) 都生成 Python 程序 (即仅 SL 推理链) 以推导答案。我们想强调以下定性差异: [4] (a) 在动机方面，我们的方法是以可解释性为驱动，而他们的方法是以性能为驱动。(b) 我们的推理链涉及对问题的结构化分解，允许没有编程背景的用户更好地理解并可能与系统互动。(c) 他们仅使用 Python 作为 SL，并且只处理数学和简单符号推理任务，而我们展示了我们的方法在多个符号语言和各种其他领域的通用性。特别是，我们创新性地将一组多样化的现实任务 (规划、多跳问答和关系推理) 重新表述为符号表示，这使我们能够通过单一框架处理它们。(d) 与之前的工作相比，我们进行了更全面的分析，特别是对推理链正确性的人工评估。

3 Method

Our method, Faithful CoT, is a 2-stage pipeline, as seen in Figure 2. Like previous CoT-style work, our prompt consists of(Q, C, A)triples. Notable differences lie in our unique interleaving of NL (natural language) and SL (symbolic language) in C , as well as the way we derive the final answer A .

我们的方法，Faithful CoT，是一个两阶段的流程，如图 2 所示。与之前的 CoT 风格工作类似，我们的提示由 (Q, C, A) 三元组组成。显著的区别在于我们独特的自然语言 (NL) 和符号语言 (SL) 的交错使用 C ，以及我们推导最终答案的方式 A 。

In the Translation stage, given a complex query Q in NL, we prompt an LM to translate it into a reasoning chain C , which interleaves NL comments and SL programs. The NL component decomposes the original query into multiple simpler, interdependent subproblems. Then, each subproblem is tackled in a task-dependent SL, such as Python, Data-log, or PDDL. In the Problem Solving stage, we call a deterministic external solver, e.g., a Python interpreter, a Datalog executor, or PDDL planner, depending on the task, to obtain the answer A from the reasoning chain C . As shown in Figure 3, we define C_{NL} to be the NL component (black) and C_{SL} to be the SL component (blue) in C . Though we separate the two components notationally, they are interleaved in the generation. Using this approach, C is guaranteed to be a faithful model explanation, since our final A is the result of deterministically executing C_{SL} . Moreover, C_{NL} allows the user to better understand the reasoning process.

在翻译阶段，给定一个用 NL 表达的复杂查询 Q ，我们提示一个语言模型将其翻译为推理链 C ，该推理链交错了 NL 注释和 SL 程序。NL 组件将原始查询分解为多个更简单、相互依赖的子问题。然后，每个子问题在任务相关的 SL 中处理，例如 Python、Datalog 或 PDDL。在问题解决阶段，我们调用一个确定性的外部求解器，例如 Python 解释器、Datalog 执行器或 PDDL 规划器，具体取决于任务，从推理链 C 中获得答案 A 。如图 3 所示，我们定义 C_{NL} 为 NL 组件 (黑色)，定义 C_{SL} 为 SL 组件 (蓝色) 在 C 中。尽管我们在符号上将两个组件分开，但它们在生成过程中是交错的。使用这种方法，C 被保证是一个忠实的模型解释，因为我们的最终 A 是确定性执行 C_{SL} 的结果。此外，C_{NL} 使用户能够更好地理解推理过程。[5]

We apply this method to 4 types of complex reasoning tasks: MWP, Multi-hop QA, Planning, and Relational Inference. Next, we will illustrate how our method works for each of them, with examples from Figure 3.

我们将此方法应用于 4 种复杂推理任务: 数学文字问题 (MWP)、多跳问答、规划和关系推理。接下来，我们将通过图 3 中的示例说明我们的方法如何适用于每种任务。

[4] Also see Appendix B. 3 for an empirical comparison. 另见附录 B.3 进行实证比较。

[5] While no constraints are enforced between C_{NL} and C_{SL} in our main experiments, we analyze this in Section C.4. 尽管在我们的主要实验中 C_{NL} 和 C_{SL} 之间没有强制约束，但我们在 C.4 节中对此进行了分析。

3.1 Math Word Problems (MWP) 数学文字问题

Given a grade-school math question Q written in NL ("If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?", shown in green in Figure 3), we want to obtain A as a real-valued number (5). In the Translation stage, we prompt the LM to take in Q and generate a reasoning chain C , which interleaves C_{NL} and C_{SL} . Specifically, the C_{NL} component consists of three types of information:

给定一个用 NL 表达的学龄前数学问题 Q ("如果停车场有 3 辆车，且又有 2 辆车到达，停车场里总共有多少辆车?" 如图 3 中绿色部分所示)，我们希望获得 A 作为一个实数 (5)。在翻译阶段，我们提示语言模型接受 Q 并生成一个推理链 C ，该推理链交错了 C_{NL} 和 C_{SL} 。具体而言，C_{NL} 组件由三种类型的信息组成:

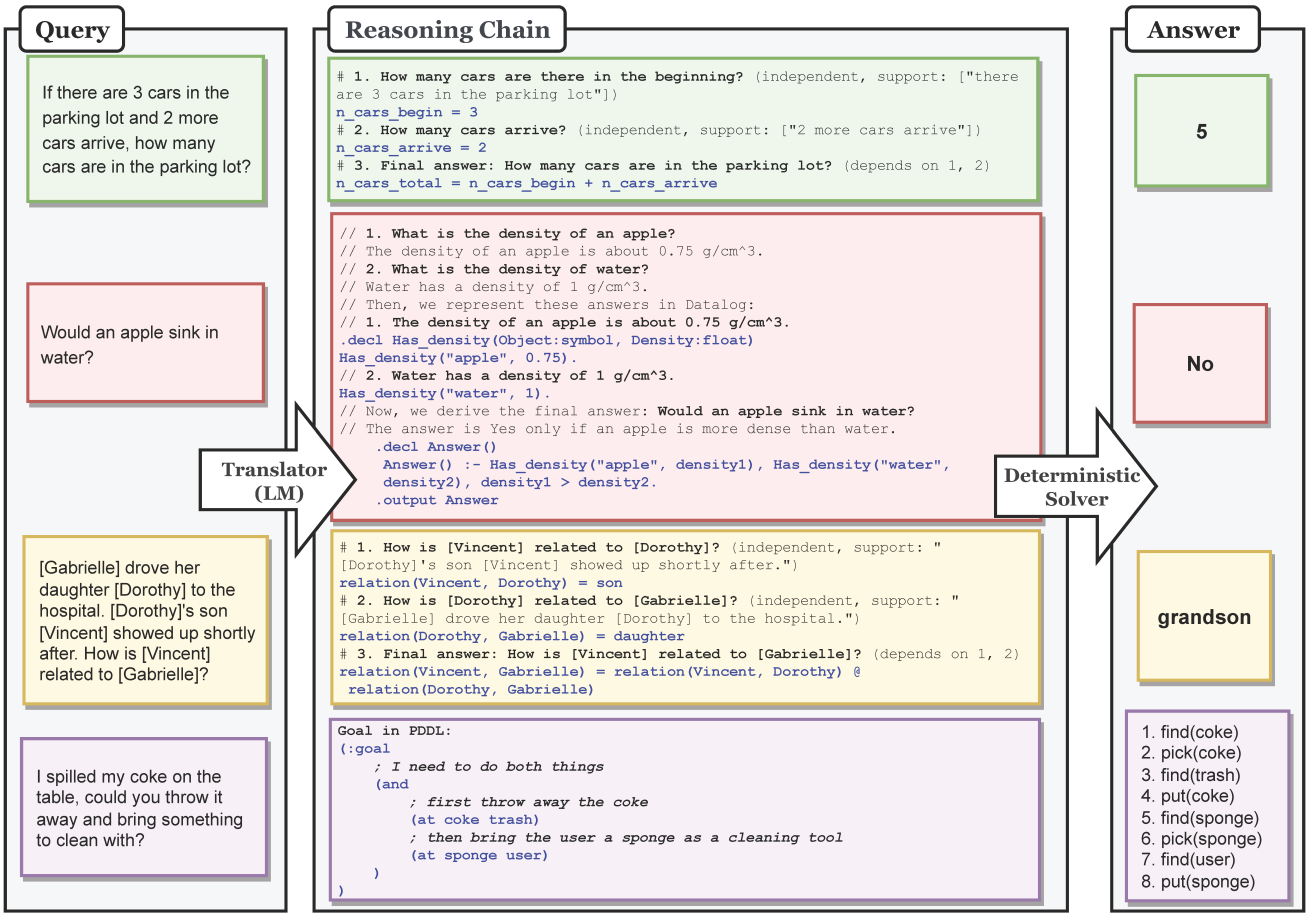

Figure 3: Examples from each task (Math Word Problems, Multi-hop QA, Relational Inference, and Planning) showing our 2-stage Translation and Problem Solving pipeline. 来自每个任务 (数学文字问题、多跳问答、关系推理和规划) 的示例，展示了我们的两阶段翻译和问题解决流程。

(a) Subquestions: Q is broken down into multiple smaller-scale subquestions, e.g., "1. how many cars are there in the beginning?", "2. how many cars arrive?", and " 3 . how many cars are in the parking lot?".

(b) Dependency Graph: Each subquestion can either be answered directly via context (subques-tions 1 and 2 are "independent") or rely on answers to previous subquestions (subquestion 3 "depends on 1 and 2").

(c) Rationales: Each subquestion is accompanied with rationale(s) to support the answer (the "support" field). The rationales can be either a subset of the original context ("2 more cars arrive") or any external knowledge ("there are 7 days in a week") relevant to the subquestion.

(a) 子问题: Q 被分解为多个小规模子问题，例如，"1. 开始时有多少辆车?"，"2. 有多少辆车到达?" 以及 "3. 停车场里有多少辆车?"。
(b) 依赖图: 每个子问题可以通过上下文直接回答 (子问题 1 和 2 是 "独立的") 或依赖于之前子问题的答案 (子问题 3 "依赖于 1 和 2")。
(c) 理由: 每个子问题都有理由来支持答案 ("支持" 字段)。这些理由可以是原始上下文的一个子集 ("再到达 2 辆车") 或与子问题相关的任何外部知识 ("一周有 7 天")。

Each subquestion and its corresponding dependencies and rationales inform the subsequent generation of C_{SL} . In our example in Figure 3, C_{SL} consists of Python code generated to answer each subquestion in C_{NL} . During the Problem Solving stage, we execute C_{SL} using our solver, a Python interpreter, to derive A (5 cars in the end).

每个子问题及其相应的依赖关系和理由为后续生成 C_{SL} 提供信息。在我们在图 3 中的示例中，C_{SL} 由生成的 Python 代码组成，用于回答 C_{NL} 中的每个子问题。在问题解决阶段，我们使用求解器 (一个 Python 解释器) 执行 C_{SL} ，以推导出 A (最终 5 辆车)。

3.2 Multi-hop QA **多跳问答**

Given a complex question Q that involves multiple steps of reasoning (e.g., "Would a pear sink in water?", shown in red in Figure 3), we want to obtain the answer A as a Boolean value or string value variable. Similar to our MWP task formulation, C interleaves C_{NL} (NL comments), and C_{SL} (symbolic program). Depending on the nature of the task, the format of the reasoning chain C is slightly different: for some datasets, the LM first generates all subquestions and their answers in NL, and then represents these answers as SL to derive A (see Figure 3); for others, the LM interleaves the NL subquestions and the SL program, similar to the case of MWP (see Table 14 and Table 15 for examples). In terms of SL, we use both Python and Datalog, also depending on the dataset. As Multi-hop QA problems involve multi-step reasoning to solve, C_{SL} often utilizes Boolean algebra and string comparisons (in Python) along with relation definitions and logic programming (in Data-log). We use their corresponding interpreter as our deterministic solver to execute C_{SL} and obtain A .

给定一个复杂的问题 Q ，涉及多个推理步骤 (例如，“梨会沉入水中吗?” 在图 3 中以红色显示)，我们希望将答案 A 作为布尔值或字符串值变量。类似于我们的 MWP 任务表述，C 交错了 C_{NL} (自然语言评论) 和 C_{SL} (符号程序)。根据任务的性质，推理链 C 的格式略有不同: 对于某些数据集，语言模型首先生成所有子问题及其在自然语言中的答案，然后将这些答案表示为符号语言以推导出 A (见图 3)；对于其他数据集，语言模型交错自然语言子问题和符号语言程序，类似于 MWP 的情况 (见表 14 和表 15 的示例)。在符号语言方面，我们使用 Python 和 Datalog，这也取决于数据集。由于多跳问答问题涉及多步骤推理，因此 C_{SL} 通常利用布尔代数和字符串比较 (在 Python 中)，以及关系定义和逻辑编程 (在 Datalog 中)。我们使用相应的解释器作为确定性求解器来执行 C_{SL} 并获得 A 。

In the example from Figure 3, the LM first generates the subquestions, "1. What is the density of a pear?" and "2 . What is the density of water?", which are individually answered in NL. The answers ("Water has a density of $1g/\text{cm}^{3,}$ ") are converted to Datalog statements (Has_density("water", 1)), which are then combined to formalize the truth condition of the final answer. Finally, we execute the Datalog program to determine that a pear would not sink in water.

在图 3 的示例中，语言模型首先生成子问题，“1. 梨的密度是多少?” 和 “2. 水的密度是多少?”，这些问题分别用自然语言回答。答案 (“水的密度是 $1g/\text{cm}^{3,}$ ”) 被转换为 Datalog 语句 (Has_density("water", 1))，然后将这些语句组合以形式化最终答案的真值条件。最后，我们执行 Datalog 程序以确定梨不会沉入水中。

3.3 Planning **规划**

In a user-robot interaction scenario, given a household task query Q from a user, we want to come up with a plan of actions A that the robot should take in order to accomplish the task. For example, in Figure 3, given user query "I spilled my coke on the table, could you throw it away and bring something to clean with?", a possible plan can be "find(coke), pick(coke), find(trash), put(coke) ...". In the Translation stage, an LM translates Q into C , consisting of C_{NL} (which breaks down Q into subtasks) and C_{SL} (which represents the subtasks as a symbolic goal in PDDL - a language to define and solve classical planning problems). Figure 3 shows this translation, with C_{SL} in blue and C_{NL} in black. Finally, we call a PDDL Planner as the deterministic solver to obtain A , a plan to accomplish the goal C_{SL} under the predefined scenario.

在用户与机器人交互的场景中，给定用户的家庭任务查询 Q ，我们希望提出一个机器人应采取的行动计划 A 以完成任务。例如，在图 3 中，给定用户查询 "我把可乐洒在桌子上了，你能把它扔掉并带来一些清洁工具吗?"，一个可能的计划可以是 "find(coke), pick(coke), find(trash), put(coke) ..."。在翻译阶段，语言模型将 Q 翻译为 C ，由 C_{NL} (将 Q 分解为子任务) 和 C_{SL} (将子任务表示为 PDDL[6] 中的符号目标——一种定义和解决经典规划问题的语言) 组成。图 3 显示了这种翻译，其中 C_{SL} 用蓝色表示，C_{NL} 用黑色表示。最后，我们调用 PDDL 规划器作为确定性求解器，以获得 A ，这是在预定义场景下实现目标 C_{SL} 的计划。

3.4 Relational Inference **关系推理**

Given a relational inference problem Q written in NL , we want to obtain A as a string-valued variable. For example, the CLUTRR (Sinha et al., 2019) dataset involves inferring the family relationship (e.g., "grandson") between two people from a short story (e.g., "[Gabrielle] drove her daughter [Dorothy] to the hospital. [Dorothy]'s son [Vincent] showed up shortly after. How is [Vincent] related to [Gabrielle]?", shown in yellow in Figure 3). During the Translation stage, we prompt the LM to generate C , consisting of C_{NL} and C_{SL} . Similar to previous tasks, C_{NL} breaks down Q into sub-questions ("How is [Vincent] related to [Dorothy]" and "How is [Dorothy] related to [Gabrielle]"), as well as provide input extracts as rationales to support the answer ("[Dorothy]'s son [Vincent] showed up shortly after", etc.). Each subquestion in C_{NL} is answered in C_{SL} via a relational expression representing the relation between the mentioned entities, for example, relation(Vincent, Dorothy)=son denotes that Vincent is Dorothy's son. In the Problem Solving stage, our solver is a simple relational inference engine that relies on a set of transitivity rules provided by Zhang et al. (2022) among possible family relationships, e.g., son@daughter=grandson (the son of one's daughter is one's grandson). Our solver recursively applies these rules on C_{SL} to derive A , and determine that Vincent is Gabrielle's grandson.

给定一个关系推理问题 Q 用 NL 表示，我们希望获得 A 作为字符串值变量。例如，CLUTRR 数据集 (Sinha et al., 2019) 涉及从短故事中推断两个人之间的家庭关系 (例如，"孙子")，如 "[Gabrielle] 驾车带她的女儿 [Dorothy] 去医院。[Dorothy] 的儿子 [Vincent] 不久后出现。[Vincent] 与 [Gabrielle] 的关系是什么?" (如图 3 中黄色部分所示)。在翻译阶段，我们提示大型语言模型生成 C ，由 C_{NL} 和 C_{SL} 组成。与之前的任务类似，C_{NL} 将 Q 分解为子问题 ("[Vincent] 与 [Dorothy] 的关系是什么?" 和 "[Dorothy] 与 [Gabrielle] 的关系是什么?")，并提供输入摘录作为支持答案的理由 ("[Dorothy] 的儿子 [Vincent] 不久后出现" 等)。每个子问题在 C_{NL} 中通过表示提到的实体之间关系的关系表达式在 C_{SL} 中得到回答，例如，relation(Vincent, Dorothy)=son 表示 Vincent 是 Dorothy 的儿子。在问题解决阶段，我们的求解器是一个简单的关系推理引擎，依赖于 Zhang et al. (2022) 提供的一组关于可能家庭关系的传递规则，例如，son@daughter=grandson(一个人的女儿的儿子是一个人的孙子)。我们的求解器递归地在 C_{SL} 上应用这些规则以推导 A ，并确定 Vincent 是 Gabrielle 的孙子。

[6] https://en.wikipedia.org/wiki/Planning_ Domain_Definition_Language. A goal is a special construct in PDDL. 目标是 PDDL 中的一个特殊构造。

4 Experimental setup 实验设置

4.1 Datasets

Here, we summarize the evaluation datasets used for each domain. We select the same number (6 to 10, depending on the task) of exemplars as in Wei et al. (2022) to form our few-shot prompt, which can be found in our repository. Unless otherwise stated, we use the official splits: training set for exemplar selection, validation set for prompt tuning, and test set for evaluation.

在这里，我们总结了用于每个领域的评估数据集。我们选择与 Wei et al. (2022) 相同数量 (6 到 10，具体取决于任务) 的示例来形成我们的少量示例提示，这些示例可以在我们的仓库中找到。除非另有说明，我们使用官方划分: 训练集用于示例选择，验证集用于提示调整，测试集用于评估。[7]

Math Word Problems (MWP). We follow Wei et al. (2022) and consider the same five MWP benchmarks: GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), MultiArith (Roy and Roth, 2015), ASDiv (Miao et al., 2020), and AQuA (Ling et al., 2017). For all datasets, the input question is phrased in NL. The answer is a string-valued mathematical expression for AQuA, and one or more integer(s) for all other datasets. We use the same 8-shot prompt for all datasets except AQuA. Multi-hop QA. We consider the three datasets: StrategyQA (Geva et al., 2021), a dataset of open-domain questions that require an implicit multistep strategy to answer, e.g., "Did Aristotle use a laptop?" involves answering "1. When did Aristotle live?", "2. When was the laptop invented?", and "3. Is #2 before #1?"; Date Understanding from BIG-bench (BIG-Bench collaboration, 2021), which asks the model to infer a date from a context, by performing computation on relative periods of time; and finally, Sports Understanding from BIG-bench, which asks the model to decide whether an artificially constructed statement related to sports is plausible or implausible. Since the latter two datasets do not have a training set, we follow Wei et al. (2022) and select 10 examples from the test set to form the prompt and use the rest for evaluation.

数学文字问题 (MWP)。我们遵循 Wei et al. (2022) 的方法，考虑相同的五个 MWP 基准:GSM8K (Cobbe et al., 2021)、SVAMP (Patel et al., 2021)、MultiArith (Roy and Roth, 2015)、ASDiv (Miao et al., 2020) 和 AQuA (Ling et al., 2017)。对于所有数据集，输入问题以自然语言表述。对于 AQuA，答案是一个字符串值的数学表达式，而对于所有其他数据集，答案是一个或多个整数。我们对除 AQuA 外的所有数据集使用相同的 8-shot 提示。多跳问答。我们考虑三个数据集:StrategyQA (Geva et al., 2021)，这是一个开放领域问题的数据集，需要隐含的多步骤策略来回答，例如，"亚里士多德使用过笔记本电脑吗?" 涉及回答 "1. 亚里士多德生活在什么时候?" "2. 笔记本电脑是什么时候发明的?" 和 "3. #2 在 #1 之前吗?"；来自 BIG-bench 的日期理解 (BIG-Bench collaboration, 2021)，要求模型从上下文中推断一个日期，通过对相对时间段进行计算；最后，来自 BIG-bench 的体育理解，要求模型判断与体育相关的人工构造的陈述是否合理或不合理。由于后两个数据集没有训练集，我们遵循 Wei et al. (2022) 的方法，从测试集中选择 10 个示例来形成提示，其余用于评估。

Planning. We use the SayCan dataset (Ahn et al., 2022), which assumes a scenario of a robot operating in a kitchen, helping the user with household tasks, e.g., "bring a coke to the table". There are a number of locations and objects that the robot can interact with. The robot can only perform a fixed set of actions, including find, pick, and put. The task is to map a user query in NL to a plan of predefined actions. Following Wei et al. (2022), we manually write 7 exemplars, since no training set is provided.

规划。我们使用 SayCan 数据集 (Ahn et al., 2022)，假设一个机器人在厨房中操作，帮助用户完成家务任务，例如 "把可乐拿到桌子上"。机器人可以与多个位置和物体进行交互。机器人只能执行一组固定的动作，包括寻找、拾取和放置。任务是将用户的自然语言查询映射到预定义动作的计划中。根据 Wei et al. (2022)，我们手动编写了 7 个示例，因为没有提供训练集。

[7] See Appendix E for dataset statistics, examples, data cleaning method, splits, prompt construction strategy, etc. 请参见附录 E 以获取数据集统计信息、示例、数据清理方法、拆分、提示构建策略等。

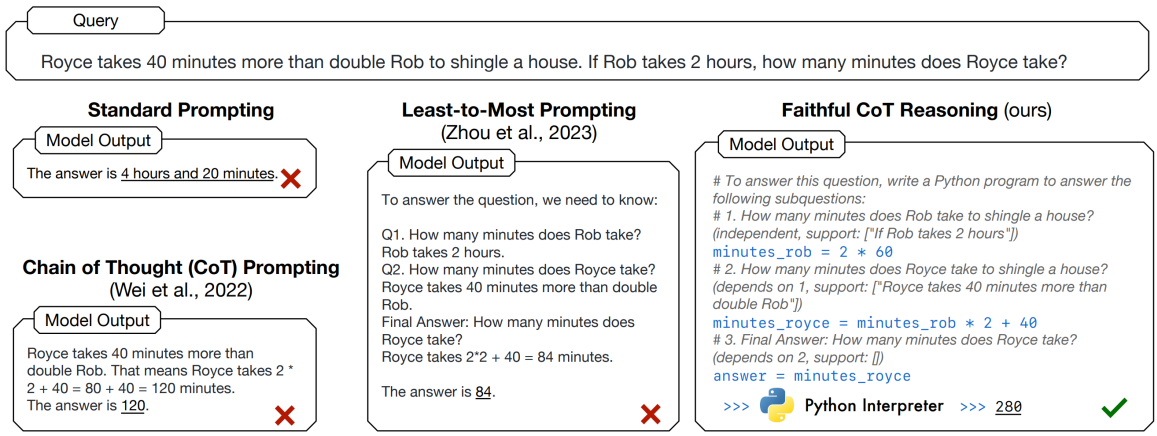

Figure 4: A sample output for a math question from three baselines and Faithful CoT (our method). The ground-truth answer is 280, and only our method correctly computes the answer.

图 4: 来自三个基线和 Faithful CoT(我们的方法) 的数学问题的示例输出。真实答案是 280，只有我们的方法正确计算了答案。

Relational inference. We use the CLUTRR (Sinha et al., 2019) benchmark described in Section 3.4. The dataset has multiple splits based on the number of intermediate steps K required to reach the answer. We construct the prompt using 8 exemplars with $K \in \{2, 3\}$, and test the models on the remaining examples with K up to 10 .

关系推理。我们使用第 3.4 节中描述的 CLUTRR (Sinha et al., 2019) 基准。该数据集根据达到答案所需的中间步骤数量 K 进行了多次拆分。我们使用 8 个示例构建提示，并在剩余示例上测试模型，最多 K 10。

4.2 Evaluation Metrics **评估指标**

We evaluate the model performance with final answer accuracy as the main metric. Following previous work (Wei et al., 2022; Wang et al., 2022; Chen et al., 2022), for all MWP datasets (except AQuA) where the answer contains integer(s), a correct answer is defined as the exact match between the prediction and the ground truth both converted to the nearest integer; for StrategyQA and Sports Understanding where the answer is a Boolean value, it is defined as the exact match between the prediction and the ground truth both evaluated as a Boolean variable; for SayCan, the generated plan is considered correct if it is among the ground truth plans; for all other datasets, we rely on the exact match between the prediction string and the ground truth string. Additionally, we evaluate the human-rated plausibility of the reasoning chain in Appendix D.

我们通过最终答案的准确性来评估模型性能，这是主要指标。根据之前的工作 (Wei et al., 2022; Wang et al., 2022; Chen et al., 2022)，对于所有 MWP 数据集 (除了 AQuA)，如果答案包含整数，则正确答案被定义为预测值和真实值之间的精确匹配，二者均转换为最接近的整数；对于 StrategyQA 和 Sports Understanding，其中答案是布尔值，正确答案被定义为预测值和真实值之间的精确匹配，二者均被评估为布尔变量；对于 SayCan，如果生成的计划在真实计划中，则被认为是正确的；对于所有其他数据集，我们依赖于预测字符串和真实字符串之间的精确匹配。此外，我们在附录 D 中评估推理链的人类评分的合理性。

4.3 Baselines

We compare our method to three other few-shot prompting baselines, shown in Figure 4: standard prompting, popularized by Brown et al. (2020), with demonstrations of only the question and the answer; CoT (Wei et al., 2022), which additionally includes an NL reasoning chain; and Least-to-Most (LtM) (Zhou et al., 2022), which decomposes the question in NL but does not involve SL. All prompting methods are compared under two decoding strategies: greedy decoding, where the LM samples the most probable next token from the vocabulary (i.e., temperature = 0.0); and self-consistency decoding (Wang et al., 2022), where the LM generates multiple reasoning chains and chooses the final chain based on majority voting on the evaluated answer (we use a temperature of 0.4 and 40 generations for all datasets). We reproduce the baseline results ourselves in cases when they are not reported on certain tasks or when we clean the test set.

我们将我们的方法与其他三种少量提示基线进行比较，如图 4 所示: 标准提示，由 Brown et al.(2020) 推广，仅包含问题和答案的示例；CoT(Wei et al., 2022)，此外还包括一个 NL 推理链；以及 Least-to-Most (LtM)(Zhou et al., 2022)，它在 NL 中分解问题，但不涉及 SL。所有提示方法在两种解码策略下进行比较: 贪婪解码，其中 LM 从词汇中采样最可能的下一个标记 (即，温度 = 0.0)；以及自一致性解码 (Wang et al., 2022)，其中 LM 生成多个推理链，并根据对评估答案的多数投票选择最终链 (我们对所有数据集使用 0.4 的温度和 40 次生成)。[8] 在某些任务未报告基线结果或我们清理测试集时，我们自己重现基线结果。

4.4 LMs

We use OpenAI Codex (Chen et al., 2021) (code-davinci-002) in Section 5 and experiment with four other code-generation LMs in Appendix C.3.

我们在第 5 节中使用 OpenAI Codex(Chen et al., 2021)(code-davinci-002)，并在附录 C.3 中实验四种其他代码生成 LMs。

5 Results

Our results on all datasets are shown in Table 1. With code-davinci-002 as the Translator, Faithful CoT outperforms all baselines across the vast majority of datasets and domains under both decoding strategies. With greedy decoding, Faithful CoT outperforms all baselines on 8 of the 10 datasets, by a relative improvement of up to 14.2% on MWP, 3.4% on Planning, 6.5% on Date Understanding from Multi-hop QA, and a surprising 21.4% on Relational Inference. Generally, we see larger gains on harder datasets. Take MWP as an example: on simpler datasets where CoT already performs decently (e.g., MultiArith and AsDiv, where most questions require only 1-2 steps to solve), the gains are smaller (0.3% to 2.4%); however, we see the largest gain (14%) on the most difficult GSM8K, which requires up to 8 steps to solve. With self-consistency decoding, Faithful CoT still performs the best on 7 out of the 10 datasets. Compared to greedy decoding, the relative gain increases on 4 datasets (AQUA: 12.1% → 18.1%, SayCan: 3.4% → 5.5% , Date Understanding: 6.5% → 10.8% , and CLUTRR: 21.4% → 41.3%), but decreases or remains unchanged for the remaining three MWP datasets (GSM8K: 9.0% → 2.6%, SVAMP: 3.9% → 2.3% , ASDiV 0.2% → 0.2%) .

我们在所有数据集上的结果如表 1 所示。以 code-davinci-002 作为翻译器，Faithful CoT 在绝大多数数据集和领域下的两种解码策略中均优于所有基线。在贪婪解码下，Faithful CoT 在 10 个数据集中的 8 个上表现优于所有基线，在 MWP 上相对提升高达 14.2%，在规划上提升 3.4%，在多跳问答中的日期理解上提升 6.5%，在关系推理上更是惊人地提升了 21.4%。一般来说，我们在更难的数据集上看到更大的收益。以 MWP 为例: 在 CoT 已经表现良好的简单数据集上 (例如 MultiArith 和 AsDiv，其中大多数问题只需 1-2 步即可解决)，收益较小 (0.3% 到 2.4%)；然而，我们在最困难的 GSM8K 上看到最大的收益 (14%)，该数据集需要最多 8 步才能解决。在自一致性解码下，Faithful CoT 在 10 个数据集中的 7 个上仍然表现最佳。与贪婪解码相比，4 个数据集的相对收益增加 (AQUA: 12.1% → 18.1%，SayCan: 3.4% → 5.5% ，日期理解: 6.5% → 10.8% ，以及 CLUTRR: 21.4% → 41.3%)，但在其余三个 MWP 数据集上 (GSM8K: 9.0% → 2.6%，SVAMP: 3.9% → 2.3% ，ASDiV 0.2% → 0.2%)) 则减少或保持不变。

[8] Note that we do not report the performance of standard prompting with self-consistency decoding, since when the number of sampled outputs is large enough, this converges to standard prompting with greedy decoding (Wang et al., 2022). 请注意，我们没有报告标准提示与自一致性解码的性能，因为当采样输出的数量足够大时，这会收敛到与贪婪解码的标准提示 (Wang et al., 2022)。

Method	MathWordProblems					Planning SayCan	Multi-hop QA			Relation
	GSM8K	SVAMP	MultiArith	ASDiv	AQuA		StrategyQA	Date	Sport	CLUTRR
Greedy Decoding										
Standard	19.6	69.5	43.8	72.1	31.5	82.5	63.9	51.3	71.9	42.0
CoT	63.3	77.3	96.5	80.0	42.1	86.4	72.5	59.9	98.6	48.5
LtM	38.3	80.3	74.0	76.5	40.6	77.7	72.2	76.6	99.5	47.2
Faithful CoT (ours)	72.3	83.4	98.8	80.2	47.2	89.3	63.0	81.6	99.1	58.9
Self-Consistency Decoding										
CoT	78.0	86.8	100.0	84.2	52.0	89.3	79.8	63.8	98.0	45.7
LtM	38.8	80.5	74.0	76.3	44.9	76.7	71.9	77.2	99.4	50.9
Faithful CoT (ours)	80.0	88.8	99.2	84.4	61.4	94.2	65.2	85.5	99.0	71.9

Table 1: Accuracy of different prompting methods on 10 reasoning datasets from 4 domains. We compare our method, Faithful CoT, with standard (Brown et al., 2020), CoT (Wei et al., 2022), and Least-to-Most prompting (Zhou et al., 2022), with code-davinci-002 as the LM. The best results within each decoding strategy are bolded.

表 1: 来自 4 个领域的 10 个推理数据集上不同提示方法的准确性。我们将我们的方法 Faithful CoT 与标准方法 (Brown et al., 2020)、CoT(Wei et al., 2022) 和 Least-to-Most 提示 (Zhou et al., 2022) 进行比较，使用 code-davinci-002 作为语言模型。每种解码策略中的最佳结果用粗体显示。

On the other hand, we do not see clear empirical gains on two multi-hop QA datasets, Sports Understanding on StrategyQA. On Sports Understanding, Faithful CoT and LtM both have near perfect accuracy (99+), suggesting that the dataset is almost saturated. On StrategyQA, however, the performance of our method is still far from the baselines. To understand why, we specifically compare the examples where CoT makes a correct prediction but our method fails. As shown in Figure 11 in Appendix F, we find that the likely primary cause is the sparsity of Datalog in the pretraining data for Codex, as an overwhelming 29% of errors are syntax-related. Moreover, including Datalog in the prompt also interferes with NL generation, making it harder for Codex to produce relevant subques-tions (17%), retrieve knowledge correctly (10%), and come up with valid reasoning from the knowledge to the answer (10%). Another potential cause is the nature of the task, as the difficulty for many StrategyQA questions does not lie in reasoning but rather in knowledge retrieval, which makes the advantages of our deterministic solver less obvious. Still, with further pretraining on Datalog, we believe that there is room for improvement.

另一方面，我们在两个多跳问答数据集 Sports Understanding 和 StrategyQA 上没有看到明显的经验性提升。在 Sports Understanding 上，Faithful CoT 和 LtM 的准确率几乎完美 (99+)，这表明该数据集几乎饱和。然而，在 StrategyQA 上，我们的方法的表现仍然远低于基线。为了理解原因，我们特别比较了 CoT 做出正确预测但我们的方法失败的示例。如附录 F 中的图 11 所示，我们发现主要原因可能是 Codex 预训练数据中 Datalog 的稀疏性，因为高达 29% 的错误与语法相关。此外，在提示中包含 Datalog 也干扰了自然语言生成，使得 Codex 更难生成相关的子问题 (17%)、正确检索知识 (10%) 以及从知识推导出有效的推理到答案 (10%)。另一个潜在原因是任务的性质，因为许多 StrategyQA 问题的难点不在于推理，而在于知识检索，这使得我们确定性求解器的优势不那么明显。尽管如此，我们相信通过进一步在 Datalog 上进行预训练，仍然有改进的空间。

To see how generalizable our method is, we also experiment with four alternative LMs and observe consistent gains brought by Faithful CoT over the baselines, as shown in Appendix C.3. In particular, with GPT-4, we set the new few-shot SOTA results on 7 datasets, achieving 95.0+ accuracy in four out of five MWP and two out of three Multi-hop QA datasets. Overall, these results suggest that faithfulness does empirically improve performance.

为了查看我们的方法的普适性，我们还实验了四种替代语言模型，并观察到 Faithful CoT 相对于基线的一致提升，如附录 C.3 所示。特别是，使用 GPT-4，我们在 7 个数据集上设定了新的少样本 SOTA 结果，在五个 MWP 数据集中有四个和三个多跳问答数据集中有两个达到了 95.0+ 的准确率。总体而言，这些结果表明，忠实性确实在经验上提高了性能。

6 Analysis

In this section, we perform an extensive analysis of the strengths and weaknesses of our method, to better understand the role of different components, the robustness to design choices, the plausibility of generated reasoning chains, as well as frequent error patterns where it still struggles. Here, we only show the first two aspects; see the rest in Appendix C. Unless otherwise stated, we choose one dataset from each domain (GSM8K, Date Understanding, Say-Can, and CLUTRR) and use code-davinci-002 outputs with greedy decoding.

在本节中，我们对我们方法的优缺点进行了广泛分析，以更好地理解不同组件的作用、对设计选择的鲁棒性、生成推理链的合理性以及仍然存在的常见错误模式。在这里，我们仅展示前两个方面；其余内容见附录 C。除非另有说明，我们从每个领域 (GSM8K、日期理解、Say-Can 和 CLUTRR) 选择一个数据集，并使用 code-davinci-002 的贪婪解码输出。

6.1 Ablation Study

Given the strong performance of Faithful CoT, we now address a natural question: how much does each part of the prompt contribute to the accuracy? We perform an ablation study where we remove different parts of the prompt and see how the performance changes. In addition to the original prompt ("Full"), we test four variations, illustrated with the example from Figure 4:

鉴于 Faithful CoT 的强大表现，我们现在提出一个自然的问题: 提示的每个部分对准确性贡献多少？我们进行了一项消融研究，去除提示的不同部分，观察性能变化。除了原始提示（“完整”）外，我们测试了四种变体，示例来自图 4:

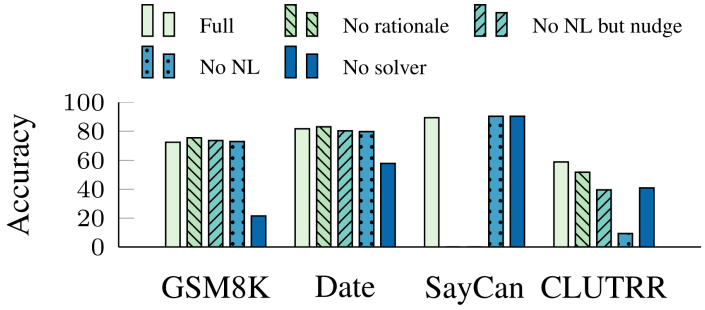

Figure 5: Ablation study results: accuracy when we remove different parts of the prompt. See Section 6.1 for details. 消融研究结果: 当我们去除提示的不同部分时的准确性。详细信息见第 6.1 节。

No rationale. We remove the rationales, i.e., everything in the brackets from the NL comments, e.g., "independent, support: ['There are 15 trees']".

无理由我们去除理由，即 NL 评论中括号内的所有内容，例如，“独立，支持:[' 有 15 棵树']”。

No NL but nudge. We remove all NL comments except the "nudge" line: e.g., "# To answer this question, we write a Python program to answer the following subquestions".

无 NL **但有提示**我们去除所有 NL 评论，除了“提示”行: 例如，“# 为了回答这个问题，我们编写一个 Python 程序来回答以下子问题”。

No NL. We remove all NL comments.

无 NL 我们去除所有 NL 评论。

No solver. Instead of calling the external solver, we add "Answer: {answer}" to the end of every exemplar and let the LM predict the answer itself.

无求解器我们不调用外部求解器，而是在每个示例的末尾添加“答案:{答案}”，让大型语言模型自行预测答案。

Figure 5 shows the results of all prompt variations. On GSM8K, Date Understanding, and Say-Can, NL comments contribute little to the performance, and sometimes even slightly hurt it. On CLUTRR, however, their role is crucial, since the exclusion of each component (rationale, nudge, subquestions) results in a clear accuracy drop. In particular, comparing No NL but nudge and No NL, the nudge line itself brings a striking improvement by 31.3 points.

图 5 显示了所有提示变体的结果。在 GSM8K、日期理解和 Say-Can 上，NL 评论对性能贡献不大，有时甚至会稍微影响它。然而，在 CLUTRR 上，它们的作用至关重要，因为每个组件 (理由、提示、子问题) 的排除都会导致明显的准确率下降。特别是，比较没有 NL 但有提示和没有 NL，提示线本身带来了 31.3 分的显著提升。

The external solver relieves the burden of problem solving from the LM. Without it, the accuracy suffers a huge decline on GSM8K, Date Understanding, and CLUTRR (-50.8, -22.9, and -19.4 respectively), while on SayCan it improves by 2.9 nonetheless. One potential influencing factor is that SayCan might be too homogeneous, as it contains a set of only 3 predefined actions. This can make the task relatively easy, which allows all model variants to achieve around 90% accuracy and renders the solver unnecessary. Another potential reason is the level of correspondence between the final answer and the reasoning chain for different datasets: as shown in Figure 3, the answer in SayCan is a sequence of actions (e.g., find(redbull)), each directly corresponding to one step in the reasoning chain (e.g., at redbull trash). However, the answer in the other three datasets is only a single number or string, which can only be derived after executing all the steps in the reasoning chain. Therefore, the latter type of tasks further necessitates the presence of an external solver.

外部求解器减轻了 LM 的问题解决负担。没有它，GSM8K、日期理解和 CLUTRR 的准确率大幅下降 (分别为 -50.8、-22.9 和 -19.4)，而在 SayCan 上尽管有所改善，仍然提高了 2.9。一个潜在的影响因素是 SayCan 可能过于同质化，因为它只包含 3 个预定义动作。这使得任务相对简单，使所有模型变体都能达到约 90% 的准确率，从而使求解器变得不必要。另一个潜在原因是不同数据集的最终答案与推理链之间的对应程度: 如图 3 所示，SayCan 中的答案是一系列动作 (例如，find(redbull))，每个动作直接对应推理链中的一步 (例如，在 redbull 垃圾桶)。然而，其他三个数据集中的答案仅是一个单一的数字或字符串，只有在执行完推理链中的所有步骤后才能得出。因此，后者类型的任务进一步需要外部求解器的存在。

Exemplars	GSM8K	Date	SayCan	CLUTRR
Set 0 (Table 1)	72.3	81.6	89.3	58.9
Set 1	72.6	81.3	91.3	59.0
Set 2	71.1	85.0	85.4	57.2
Set 3	72.3	82.5	88.3	58.0
Set 4	71.2	77.4	88.3	55.5
Set 5	71.5	85.0	89.3	56.0
Mean	71.8	82.1	88.7	57.4
Std	0.6	2.8	1.9	1.5

Table 2: Robustness to the choice of exemplars.

表 2: 对示例选择的鲁棒性。

6.2 Robustness to Exemplars **对示例的鲁棒性**

We now answer the next question: how much does the choice of exemplars matter? To do this, we annotate 20 examples in total, randomly sample k (7-10, depending on the dataset) to construct the prompt, and repeat the process five times. Table 2 shows the performance of all six runs, including the original (from Table 1). The mean accuracy is close to the original (-1.5 to +1.2), still above the baselines by a large margin (7 to 17) on all datasets except the arguably easiest SayCan, considering the standard deviation (1.3 to 2.9). This strongly suggests that the benefits of Faithful CoT are minimally influenced by the choice of exemplars.

我们现在回答下一个问题: 示例的选择有多重要? 为此，我们总共注释了 20 个示例，随机抽样 k (7-10，具体取决于数据集) 以构建提示，并重复该过程五次。表 2 显示了所有六次运行的性能，包括原始结果 (来自表 1)。平均准确率接近原始结果 (-1.5 到 +1.2)，在所有数据集中仍然大幅高于基线 (7 到 17)，除了可以说是最简单的 SayCan，考虑到标准差 (1.3 到 2.9)。这强烈表明，Faithful CoT 的好处受到示例选择的影响很小。

6.3 Human Evaluation of Plausibility **对合理性的人工评估**

Our main experiments use final answer accuracy as the performance measure, but this does not necessarily correspond to the validity of the reasoning chain. Technically, a model can sometimes accidentally arrive at the correct answer with an invalid reasoning chain. We then ask: when the answer is correct, how often is the reasoning chain truly correct? In other words, we want to evaluate the plausibility of the reasoning chains.

我们的主要实验使用最终答案准确性作为性能指标，但这并不一定对应于推理链的有效性。从技术上讲，模型有时可能会意外地通过无效的推理链得出正确答案。我们接着问: 当答案正确时，推理链的正确性有多高？换句话说，我们想评估推理链的合理性。

We conduct a human evaluation study on Prolific: given a generated reasoning chain that results in a correct answer, a crowd-worker selects whether it is A) completely correct, or, if incorrect, specify why with B) incorrect NL and/or C) incorrect SL . Alternatively, they can select D) flawed question and E) I am confused.

我们在 Prolific 上进行了一项人工评估研究: 给定一个生成的推理链，该推理链导致正确答案，众包工作者选择它是否 A) 完全正确，或者如果不正确，指定原因 B) 不正确 NL 和/或 C) 不正确 SL 。或者，他们可以选择 D) 有缺陷的问题和 E) 我感到困惑。[9]

The results of our study are shown in Figure 6 (see Table 8 in the Appendix for numerical results). For most domains, we see that annotators often find the reasoning chain fully correct - Sports, Say-Can, and MWP have a 90%+ correctness rate. To gain more insight into when the reasoning chain can be "incorrect", we perform an in-depth analysis of user annotations for the three worst-performing datasets - StrategyQA (66.7% correctness), Date (87.9% correctness), and CLUTRR (88.0% correctness). From our inspection, we find that annotators mistakenly mark a correct reasoning chain as incorrect at different rates based on the task (8.3% of the time for StrategyQA, 41.7% for Date Understanding, and 100% for CLUTRR). We find annotators are inaccurate for Date because they incorrectly believe the generated code misuses a Python library (relativedelta), or they complain that there is a better way to answer the question. For CLUTRR, annotators mark chains as incorrect due to known ambiguity in the dataset. For example, the grandmother of one's child may not necessarily be their parent, but also a parent-in-law.

我们研究的结果如图 6 所示 (有关数值结果，请参见附录中的表 8)。在大多数领域中，我们看到注释者通常认为推理链是完全正确的 - 体育、Say-Can 和 MWP 的正确率超过 90%。为了深入了解推理链何时可能是 "错误的"，我们对表现最差的三个数据集进行了用户注释的深入分析 - StrategyQA(66.7% 的正确率)、Date(87.9% 的正确率) 和 CLUTRR(88.0% 的正确率)。通过我们的检查，我们发现注释者在不同任务中错误地将正确的推理链标记为错误的比例不同 (StrategyQA 为 8.3%，Date 理解为 41.7%，CLUTRR 为 100%)。我们发现注释者对 Date 的不准确是因为他们错误地认为生成的代码错误地使用了一个 Python 库 (relativedelta)，或者他们抱怨有更好的方法来回答问题。对于 CLUTRR，注释者将链标记为错误是由于数据集中已知的模糊性。例如，一个孩子的祖母不一定是他们的父母，也可能是岳父母。

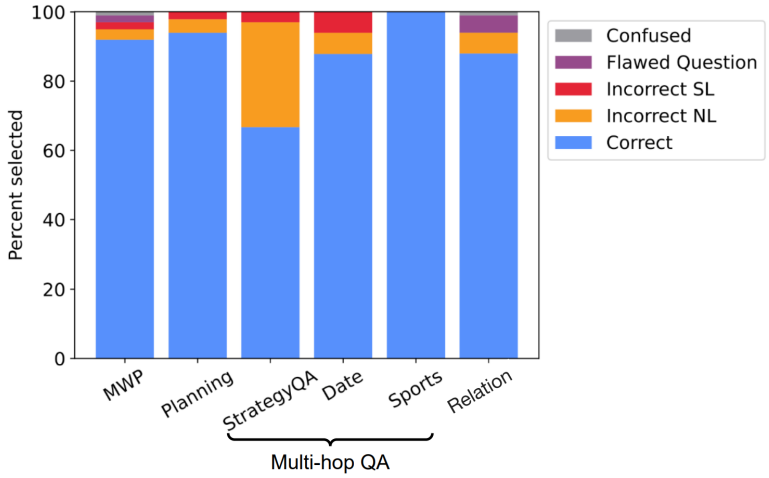

Figure 6: Human evaluation results of reasoning chain plausibility. Each column represents the percent of different answer choices selected by human evaluators in each domain/dataset. 推理链合理性的人类评估结果。每一列代表人类评估者在每个领域/数据集中选择的不同答案选项的百分比

[9] See Appendix D for more details on the human study. 有关人类研究的更多细节，请参见附录 D。

As for the remaining truly incorrect reasoning chains, we find the LM can sometimes add unnecessary steps or arrive at the correct answer by chance. The latter is especially an issue in Strate-gyQA - given that all questions have a True/False answer, it is common for an incorrect reasoning chain to result in a correct answer. For example, the LM correctly answers the question "Was Karachi a part of Alexander the Great's success?" as "True." However, the reasoning chain contains the flawed subquestion "Which countries are in Pakistan? Pakistan includes Pakistan, Afghanistan, and India." Though the final answer is correct and faithful to the LM generation, the reasoning chain contains wrong knowledge.

至于其余真正错误的推理链，我们发现大型语言模型有时会添加不必要的步骤或偶然得出正确答案。后者在 StrategyQA 中尤其成问题 - 鉴于所有问题都有一个真/假的答案，错误的推理链导致正确答案是很常见的。例如，模型正确回答了“卡拉奇是亚历山大大帝成功的一部分吗?”的问题为“是”。然而，推理链包含了有缺陷的子问题“巴基斯坦有哪些国家？巴基斯坦包括巴基斯坦、阿富汗和印度。”尽管最终答案是正确的，并且忠实于模型生成，但推理链包含错误的知识。

Overall, Faithful CoT does generate valid reasoning chains for the vast majority of the time when the answer is correct. However, we still see exceptions where the model arrives at the right answer via an incorrect reasoning chain. Though this happens infrequently, it raises concerns about when people should trust LMs. To our knowledge, we are the first to conduct a systematic human study on the plausibility of CoT-style reasoning chains, and we hope to see future work further investigate and improve on the flaws that our study brings to light.

总体而言，忠实的 CoT 确实在大多数情况下生成有效的推理链，当答案是正确时。然而，我们仍然看到例外情况，即模型通过错误的推理链得出正确答案。尽管这种情况发生得不频繁，但它引发了人们在何时应该信任大型语言模型的担忧。据我们所知，我们是首个对 CoT 风格推理链的合理性进行系统性人类研究的团队，我们希望未来的工作能进一步调查并改进我们研究所揭示的缺陷。

7 Conclusion 结论

We propose Faithful CoT, a framework that decomposes complex reasoning into Translation and Problem Solving. It guarantees that the reasoning chain is a faithful explanation of how the model arrives at the answer. We demonstrate the efficacy of our approach on 4 types of complex reasoning problems: Math Word Problems, Multi-hop QA, Planning, and Relational Inference. Our method sets new SOTA performance on 7 of the 10 datasets, while additionally providing a faithful explanation for the final answer. These results give empirical evidence that improving model interpretability, by guaranteeing the faithfulness of an explanation, does not come at the expense of overall performance; in fact, we see a strong synergy in between. Through a comprehensive analysis of the strengths and weaknesses of our method, we show its robustness to the choice of exemplars, the pivotal role of the solver, as well as frequent error patterns where it still struggles.

我们提出了 Faithful CoT，一个将复杂推理分解为翻译和问题解决的框架。它保证推理链是模型如何得出答案的真实解释。我们在四种复杂推理问题上展示了我们方法的有效性: 数学文字问题、多跳问答、规划和关系推理。我们的方法在 10 个数据集中的 7 个上设定了新的 SOTA 性能，同时为最终答案提供了真实的解释。这些结果提供了实证证据，表明通过保证解释的真实性来提高模型可解释性，并不会以整体性能为代价；事实上，我们看到两者之间存在强大的协同作用。通过对我们方法的优缺点进行全面分析，我们展示了其对示例选择的鲁棒性、求解者的关键作用，以及仍然存在的常见错误模式。

Limitations 局限性

One crucial limitation of our study is that on March 23rd, 2023, OpenAI discontinued the use of code-davinci-002. This has rendered part of our results unreplicable for any teams or researchers who have not been granted continued access to the model. This discontinuation was unexpected during our study. It raises important questions about using closed-source models for academic research.

我们研究的一个关键限制是，2023 年 3 月 23 日，OpenAI 停止使用 code-davinci-002。这使得对于未获得持续访问模型权限的团队或研究人员，我们的部分结果无法复制。这一停用在我们的研究过程中是意外的。它引发了关于使用闭源模型进行学术研究的重要问题。

Meanwhile, one methodological limitation of our approach lies in the scope of faithfulness. Currently, we guarantee that Problem Solving stage is faithful. However, the Translation stage is still opaque, meaning it is not self-interpretable how the LM generates the reasoning chain from the question. It is still an under-explored question whether it is possible to improve the interpretability of the LM generation process in general, and a few recent studies have made promising early progress (Yin and Neubig, 2022; Sarti et al., 2023) that might be used to improve the faithfulness of the Translation stage.

与此同时，我们方法的一个方法论限制在于真实性的范围。目前，我们保证问题解决阶段是可信的。然而，翻译阶段仍然不透明，这意味着 LM 如何从问题生成推理链并不自我可解释。是否可以改善 LM 生成过程的一般可解释性仍然是一个未被充分探索的问题，最近的一些研究 (Yin 和 Neubig，2022；Sarti 等，2023) 在这方面取得了有希望的早期进展，可能用于提高翻译阶段的真实性。

Finally, it still needs further exploration of the role NL comments in the reasoning chain. From our ablation study, in terms of performance, whether to include the NL comments in the reasoning chain does not make a big difference on many of the datasets, especially those where the task is not knowledge-intensive. Nevertheless, speaking of interpretability, NL comments can make the reasoning chain more structured and understandable to the end user. Further, NL comments can be an interface that allows users without a programming background to interact with and debug the model, which we leave for future work.

最后，仍需进一步探索自然语言评论在推理链中的作用。从我们的消融研究来看，在性能方面，是否在推理链中包含自然语言评论对许多数据集并没有太大区别，尤其是在任务不知识密集的情况下。然而，谈到可解释性，自然语言评论可以使推理链对最终用户更加结构化和易于理解。此外，自然语言评论可以成为一个接口，使没有编程背景的用户能够与模型互动和调试，这一点我们留待未来的工作。

Ethics Statement **伦理声明**

With the recent success of generative large LMs, they are now being used to solve complex reasoning problems. When using the output of an LM for reasoning, there is a danger that if the reasoning appears realistic, then the final answer or conclusion will also be considered reliable. As we highlighted in Figure 1 and 7, this is often not true, since an LM may produce a reasoning chain that looks plausible, but the final answer is still wrong. This work is a step in the direction of making the use of LMs more trustworthy by using the LM for just expressing its reasoning in a symbolic program and executing the program independently. In this work, we have ensured the faithfulness of the reasoning chain w.r.t how the final answer is produced in a variety of domains, but admittedly the Translation phase is still opaque. Therefore, our pipeline is still not entirely interpretable. Furthermore, as we have stressed in Section 1, faithfulness does not guarantee correctness, so our method can still sometimes produce erroneous answers, which may pose a risk for users that rely on it for decision making.

随着生成大型语言模型的近期成功，它们现在被用于解决复杂的推理问题。当使用语言模型的输出进行推理时，如果推理看起来合理，就存在最终答案或结论也会被认为可靠的危险。正如我们在图 1 和 7 中强调的，这通常并不成立，因为语言模型可能产生一个看似合理的推理链，但最终答案仍然是错误的。这项工作是朝着使语言模型的使用更加可信的方向迈出的一步，通过仅使用语言模型来表达其推理在一个符号程序中，并独立执行该程序。在这项工作中，我们确保了推理链的忠实性，关于最终答案在各种领域中的产生，但诚然，翻译阶段仍然不透明。因此，我们的流程仍然不是完全可解释的。此外，正如我们在第 1 节中强调的，忠实性并不保证正确性，因此我们的方法有时仍然会产生错误的答案，这可能对依赖它进行决策的用户构成风险。