From Good to Great: Improving Math Reasoning with Tool-Augmented Interleaf Prompting

Nuo Chen[♣] **Hongguang Li**[♢] **Baoyuan Wang**[♢] **Jia Li**[♣]

[♣]Hong Kong University of Science and Technology (Guangzhou)
Hong Kong University of Science and Technology
[♢]Xiaobing.AI
nchen022@connect.ust.hk, jialee@ust.hk

Abstract

This paper investigates the performance of Large Language Models (LLMs) and Tool-augmented LLMs in tackling complex mathematical reasoning tasks. We introduce IMR-TIP: Improving Math Reasoning with Tool-augmented Interleaf Prompting, a framework that combines the strengths of both LLMs and Tool-augmented LLMs. IMR-TIP follows the "From Good to Great" concept, collecting multiple potential solutions from both LLMs and their Tool-Augmented counterparts for the same math problem, and then selecting or re-generating the most accurate answer after cross-checking these solutions via tool-augmented interleaf prompting. The framework incorporates two key aspects: self-prompt and tool-augmented interleaf prompting (TIP). The former allows LLMs to autonomously refine and improve an initial prompt related to tool usage, while the latter enables LLMs to derive the final answer by dynamically analyzing the problem, cross-checking potential solutions, and revising previous reasoning hints in an interleaved manner. Experimental analysis shows that IMR-TIP achieves enhanced mathematical capabilities and outperforms traditional LLMs and tool-augmented LLMs in accuracy and reasoning diversity on math reasoning tasks. For instance, IMR-TIP can improve Tool-augmented ChatGPT on GSM8K-Hard from 56.0% to 65.2 %.

1 Introduction

Large language models (LLMs) (Brown et al., 2020b; Hu et al., 2021; Zeng et al., 2022; OpenAI, 2023; Chen et al., 2023a; You et al., 2022) such as GPT4 have exhibited remarkable performances across a wide array of downstream tasks. They can effortlessly tackle downstream tasks by conditioning on a scant number of in-context exemplars or plain natural language task descriptions (Brown et al., 2020a). Notwithstanding these significant advancements, even the most extensive LLMs are confronted with challenges when faced with intricate tasks that necessitate multiple reasoning steps (Gao et al., 2023; Chen et al., 2023b).

Method	Tool-Augmented LLMs			
	GSM8K	Wrong	Right	Total
LLMs-COT	Wrong	*286*	318	604
	Right	**294**	421	715
	Total	580	739	1319

Table 1: Confusion matrix between LLMs and its Tool-Augmented approach on GSM8K dataset. Here, we select the calculator as the external tool, which is implemented by `eval` function in `Python`. The experimental LLM is `text-davinci-003` (2-shot).

The capacity of LLMs to tackle complex tasks has been extensively assessed using math reasoning datasets such as GSM8K (Cobbe et al., 2021) and SVAMP (Patel et al., 2021). These datasets present math problems that cannot be directly answered but require multi-step reasoning. To encourage LLMs to engage in step-by-step reasoning, the chain-of-thought (COT) strategy (Wei et al., 2022) has emerged as the standard prompting approach. Through COT, LLMs are guided to generate a solution that consists of a sequence of intermediate steps, leading to the final answer. Nevertheless, it has been observed in previous studies that LLMs are prone to making mistakes or hallucinations, particularly during intermediate numerical computations (Qian et al., 2022; Yuan et al., 2023; Lu et al., 2022). Even a minor mistake at this stage can result in a completely incorrect final answer. In an effort to address this limitation, a series of studies (Parisi et al., 2022; Schick et al., 2023; Yao et al., 2022) have been undertaken, which leverage external tools such as the calculator to compensate for the weaknesses of LLMs. These approaches have shown significant improvements in the accuracy of answers on math reasoning tasks.

Although these approaches have led to signif-

Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ACL 2024), pages 64–79
August 15, 2024

icant improvements overall, a pertinent question arises: Can these tool-augmented LLMs outperform their traditional LLM-COT counterparts consistently? To explore this question, we conduct a pilot analysis at first: we test both LLMs-COT and its tool-augmented counterpart (calculator-augmented) on GSM8K dataset, separately. Table 1 meticulously compares the quantities of accurately and inaccurately predicted samples between the two methods. Upon careful analysis of the tabulated results, we can observe that incorporating a calculator has undeniably yielded a substantial boost (739 vs. 715) in the accuracy of LLMs on the GSM8K dataset. This improvement can be primarily attributed to the calculator's capability to mitigate the potential errors that could arise during mathematical computations within the model. Nevertheless, it is noteworthy that despite this tool augmentation, there persist cases (**294**) where the LLMs-COT outperforms its Tool-augmented LLMs counterpart, achieving accurate predictions where the latter faltered. Upon experimental analysis, the primary factors contributing to this phenomenon can be categorized as: 1) LLMs may generate different reasoning logic for the same math question due to the varying token sampling probabilities. 2) When employing tool-augmented LLMs to tackle math reasoning problems, stringent requirements are imposed on the output text to adhere to specific formats. For instance, calculator-augmented LLMs (Schick et al., 2023) must produce the output text in a predefined numerical equation format, ensuring seamless compatibility with the calculator's invocation. Conversely, program-augmented LLMs (Gao et al., 2023) necessitate the generation of specific code functions that can be executed effectively. These prescribed format requirements may influence the mathematical reasoning process of LLMs.

Based on the aforementioned analysis, we aim to capitalize on the strengths of both LLMs and Tool-augmented LLMs to further enhance their mathematical capabilities. In this work, we introduce **IMR-TIP**: **I**mproving **M**ath **R**easoning with **T**ool-augmented **I**nterleaf **P**rompting. IMR-TIP is a framework that follows the idea "*From Good to Great*" that first collects multiple potential solutions from LLMs and their Tool-Augmented approaches given the same math problem, and then selects the most correct answer or re-generates new answer after cross-checking these solutions in a tool-augmented interleaved manner. To achieve this, we design IMR-TIP in two aspects: (1) We first propose *self-prompt* to address the challenge of crafting clear, diverse and effective tool-based prompts. It allows LLMs to autonomously refine and improve an initial prompt related to tool usage, resulting in enhanced prompts. Moreover, the self-prompt can yield multiple diverse tool-based prompts through iterative runs without requiring repetitive manual modifications, eliciting varied reasoning paths (solutions) in LLMs. (2) Then given multiple solutions, we introduce *tool-augmented interleaf prompting* (TIP), a paradigm combining reasoning, acting, and external tools in an interleaved manner to arrive at the correct answer. TIP allows LLMs to dynamically analyze the problem, cross-check potential solutions, and re-evaluate even revising previous reasoning hints through *action* and *thought*. Our technical contributions are summarized as follows:

- We conduct an in-depth study for LLMs-COT and tool-augmented LLMs on the math reasoning tasks, make interesting observations and design a novel framework named IMR-TIP to improve LLMs.

- We introduce self-prompt to obtain diverse and machine-friendly tool-based prompts without manual effort.

- We carry out extensive experiments to verify that our proposed IMR-TIP significantly improves LLMs and their tool-augmented approaches on 5 math reasoning tasks based on two different backbones.

2 Methodogy

In this section, we aim to illustrate our method in detail, as seen in Figure 1 and Figure 2. We first review the problem formulation of math problem reasoning. Then we introduce our proposed self-prompt and tool-augmented interleaf prompting methods, sequentially.

2.1 Problem formulation

A math problem solving task can be defined as $\{Q, O, A\}$, where Q is the target math question, $O = \{O_1, O_2, ..., O_k\}$ are answer options if Q is a K-way multiple choice problem, A is the corresponding ground-truth answer. Given Q and O as inputs, LLMs can directly output answers or output a sequence of tokens as intermediate reasoning

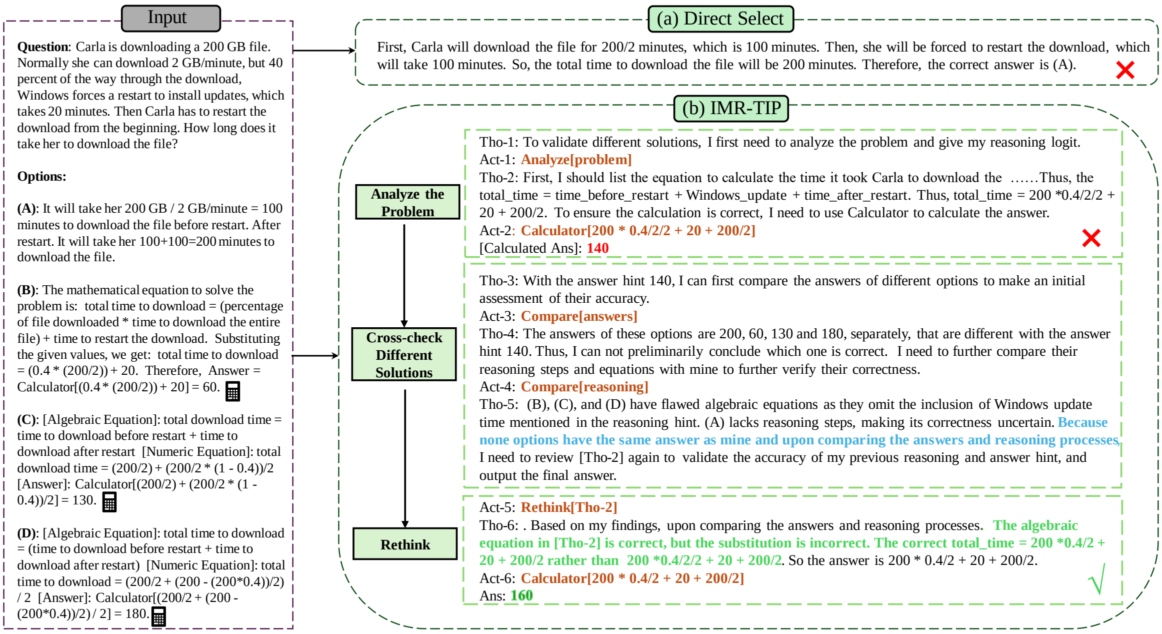

Figure 1: Comparison of 2 prompting methods: (a) *Direct Select*, which requires LLMs to select the correct one from options. (b) IMR-TIP, which derives the correct answer in an interleaved manner with *action* and *thought*. In this example, we present 3 options (Option B, C, D) from tool-augmented LLMs (using the calculator) and 1 option from LLMs COT (Option A). Tho and Act are short for Thought and Action, respectively.

steps R via COT. Then we can obtain the answer in R through regular expression matching.

2.2 IMR-TIP

The key insights of the proposed methods are two-fold: (1) We first propose self-prompt to obtain diverse tool-based prompts to induce more diverse reasoning paths and solutions. (2) We then introduce TIP to better derive the final answer from multiple reasoning solutions from LLMs and tool-augmented LLMs. The flow of our TIP can be defined as: **Initial answer determination from question analysis → Cross-Validation of options accuracy → Further verification of reasoning if discrepancies arise → Rethinking for answer accuracy if uncertain → Final answer**.

2.2.1 Self-Prompt

Crafting clear, diverse and concise tool-based prompts, which encompasses the output format, tool definition, and usage instructions, proves more challenging than traditional COT. Thus, we propose self-prompt to address this issue. Moreover, different from previous works (Wang et al., 2022; Li et al., 2023) that focus on generating k reasoning paths from one fixed prompt by sampling decoding, which limits the diversity of reasoning paths. Our proposed self-prompt can produce diverse prompts, automatically increasing prompt diversity and encouraging LLMs to think differently, eliminating the need for repetitive manual writing and editing. Concretely, we design self-prompt in three steps, as shown in Figure 2:

- Step1: We first give an initial tool-based prompt to the LLM with the instruction: "Summary the drawbacks of the current prompt and give some advice".
- Step2: With the advice and prompt problems that are outputted from LLMs, we then instruct the LLM with: "According to your advice, please rewrite the current prompt". Thus, the LLM can output the revised prompt.
- Step3: Given the revised prompt as input and ask the LLM: "Is there any problem with the revised prompt?". If the LLM responds "Yes" we will repeat steps 1 to 3 until the LLM answers "No". Otherwise, this is the final improved prompt.

Overall, our self-prompt follows the idea that "LLMs know themselves better". Through the steps mentioned above, we not only address the issue of manually crafting reasonable and clear instructions for LLMs to use tools but also enhance the diversity of prompts simultaneously via running self-prompt through multiple iterations. Experimentally, the improved tool-based prompt can help LLMs achieve better performances compared with the initial prompt. We can run it M times, resulting

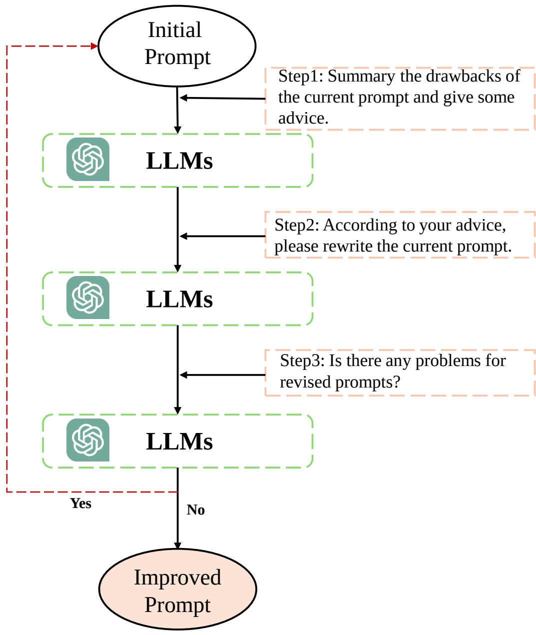

Figure 2: Overview of self-prompt.

in M diverse prompts, and then sample N reasoning paths for each prompt via sampling decoding. This way, we can obtain K = M × N tool-based diverse reasoning paths for each math problem.

2.2.2 Tool-augmented Interleaf Prompting

The core idea of TIP is: We augment the reasoning paths for each question from 1 to K+L, where L from LLMs [1] and K from their tool-augmented LLMs (obtained from self-prompt). Then we derive the final answer by observing and validating these solutions (reasoning paths) in an interleaved manner, which can be formalized as $\mathcal{TIP}(\hat{A}|Q, R_1, ...R_{K+L})$. TIP is similar to how a human brain learns when solving mathematical and multiple-choice questions: given a question and different possible solutions, analyze the problem, observe the solutions to find how the question should be solved, and then learn from these solutions, memorize or revise its own solution, conclude the answer at last.

Following this, We decompose TIP into three major steps. Given a question and multiple potential solutions: (1) ANALYZE THE PROBLEM: Before validating different answers, LLMs first need to analyze the problem and give their own reasoning logical via generating algebraic and numerical expression; (2) CROSS-CHECK DIFFERENT SOLUTIONS: Then based on the above reasoning hint, LLMs can cross-check different solutions. Considering LLMs could occasionally conclude accurate answers when generating wrong reasoning paths, we break down this into two sub-steps: compare the numerical answers and then cross-verify intermediate steps, which explores whether each solution is both sound and accurate. (3) RETHINK: Unlike traditional multiple-choice question answering tasks, where there is at least one correct solution and reliable reasoning hints, in our case, all four solutions could be incorrect, and previous reasoning hints might also be flawed. Hence, we introduce an extra step called "Rethink" which requires LLMs to review and correct their initial answer hints after observing and checking different solutions, outputting the final answer. We argue that LLMs are capable of gaining a deeper understanding of the problem by cross-checking various solutions.

Inspired by ReAct (Yao et al., 2022), we let LLMs achieve each step in an interleaved manner with *thought* and *action*. In each step, LLM receives the environmental context c_t and takes an action $a_t \in \mathcal{A}$, where $c_t = \{t_1, a_1, ..., t_{t-1}, a_{t-1}\}$ and $\mathcal{A}$ is the action space. t_1, a_1 are previous *thought* and *action* at step 1. As shown in Figure 1, there could be multiple types of useful *thoughts*, like creating action plans (*Tho-1*), reasoning process of executing the action (*Tho-2*), reasoning over the context (*Tho-6*).

Action Space As we focus on math reasoning tasks, $\mathcal{A}$ consists of the following actions to support interactive TIP: (1) *Analyze[]*, which requires analyzing the specific solution or the math problem. (2) *Compare[]*, which requires comparing and cross-checking answers or intermediate steps from different solutions or the generated answer hints. (3) *Rethink[]*, which means to double-check or review the specific thinking process in previous *thoughts*. (4) *Calculator[]*, which calls the calculator to compute the expression.

Figure 1 shows a typical example of IMR-TIP. In this example, all options are the wrong solutions, and the *direct select* and *self-consistency* approaches also give the wrong answers. In our proposed IMR-TIP, despite making an initial error in predicting during the "analyze the problem" step, the model self-corrects by cross-checking and identifying the correct equation formulation in the *Tho-2*, albeit with an algebraic mistake. In the *Tho-6*, it rectifies the error, resulting in the correct answer via calling the calculator.

Overall, our IMR-TIP has the following salient

[1] In our experiments, we obtain L reasoning paths from LLMs by sampling decoding.

features for solving math reasoning tasks: (I) Multi-view Reasoning paths: Approaching the math problem from multiple perspectives from LLMs and tool-augmented LLMs; (II) Multi-verification: Comparing and cross-checking the different solutions to a reasoning hint, including verifying the correctness of numeric answers and the intermediate steps, aiming at providing a more accurate understanding of the problem; (III) Calculation-Verification: Using the calculator to calculate the expression or verify the result; (IV) Self-Check: Re-checking the reasoning paths that include algebraic equation and substitution in previous reasoning hints after cross-checking different solutions.

In theory, M, N, K and L could be assigned higher values, resulting in more reasoning paths. However, due to the fact that these K+L solutions will be subsequently used as inputs for our TIP, we must take into account the *limitations posed by input token length and associated cost*s in the in-context learning setting. Experimentally, we have determined that setting M=3, N=1, K = M $\times$ N =3 and L=1 is appropriate.

3 Experiments

In this section, we first review our testing datasets and evaluation metrics. Then we briefly introduce our experimental settings that include backbones, and our prompts. At last, we will present our main results and ablation studies.

3.1 Datasets and Metrics

Datasets We validate our proposed approach on five math reasoning datasets: (1) **MAWPS** (Koncel-Kedziorski et al., 2016) serves as a collection of math word problems (MWPs), offering a unified testbed with 1921 samples for accessing various algorithms. (2) **SVAMP** (Patel et al., 2021), short for Simple Variations on Arithmetic Math Word Problems, is an elementary-level MWP dataset that contains 1000 examples in test set. (3) **GSM8K** (Cobbe et al., 2021) stands as a dataset comprising 1391 linguistically diverse grade school MWPs, meticulously crafted by human problem writers to ensure high quality. (4) **GSM8K-Hard** (Gao et al., 2023), an advanced iteration of GSM8K, which substitutes numbers in the questions with larger counterparts. This modification aims to assess the generalization capability of LLMs when dealing with substantial numerical values. (5) Given the limited range of three-digit numbers in the SVAMP dataset, we enhanced model assessment by randomly replacing query numbers with values between 100,000 and 10,000,000. This adjustment, while maintaining original reasoning logic, lead to the creation of **SVAMP-Hard**. SVAMP-Hard and GSM8K-Hard provide a more rigorous test of LLMs' mathematical computational capabilities.

Metrics Following previous standard works, we use the **Accuracy** as the evaluation metrics.

3.2 Experimental Settings

Backbone We conduct our experiments based on two OpenAI LLMs: GPT-3 (`text-davinci-003`) and GPT-3.5 (`gpt-3.5-turbo`) through Azure Service. We use the default parameters except temperature = 0 for greedy decoding during inference. In our experiments, we select the calculator as the external tool for tool-augmented approaches, which is implemented by Python *eval()* function.

Exemplars All quintet mathematical reasoning datasets allocate two exemplars, which are arbitrarily selected solely from the GSM8K-trainset, for both LLMs-COT and tool-enhanced LLMs. In TIP, we manually compose trajectories to use three exemplars in the prompts. Each trajectory includes multiple thought-action steps, where each thought is free-form. LLMs combine these thoughts to devise a plan ("... I first need to analyze x"), guide the reasoning process ("Option (A) is right, I then ..."), call tool ("use the calculator to compute y"), and conclude the answer ("the answer is z"). The model will end with "[Ans]" when solving the task. See Appendix A for more details about LLMs-COT, tool-augmented LLMs and TIP prompts.

Baselines We mainly compare our approach with the following baselines: (1) **COT**, which solves the problem via step-by-step reasoning with our own implementations. (2) **Tool-Augmented**, signifies LLMs employing calculators for expression computation with *initial tool-based prompts*, as elucidated in preceding sections. In this way, LLMs always need to follow a specific output format. (4) **Self-Prompt**, which utilizes the *improved too-based prompts* from self-prompt to solve math reasoning tasks. (5) **Toolformer** (Schick et al., 2023), a sophisticated paradigm that enables language models like GPT-J to utilize external tools, optimizing their performance in various tasks through fine-tuning. (4) **ART** (Paranjape et al., 2023), a framework facilitating automatic multi-step reasoning and tool-use

Backbone	Algorithms	Dataset					
		MAWPS	SVAMP	SVAMP-H	GSM8K	GSM8K-H	Average
GPT-J	Toolformer	44.0	29.4	-	-	-	-
GPT-3 (text-davinci-003)	ART	90.1	76.2	-	-	-	-
	COT	89.6	77.1	29.5	54.2	20.4	54.2
	Tool-Augmented	89.9	77.6	69.9	55.5	44.1	67.3
	Self-Prompt	90.1	78.5	70.0	56.1	44.5	67.8
	IMR-TIP	**90.7**	**79.3**	**73.7**	**61.5**	**47.4**	**70.5**
GPT-3.5 (gpt-3.5-turbo)	COT	91.0	76.5	39.2	73.0	35.0	62.9
	Tool-Augmented	91.9	77.0	78.5	75.5	54.0	75.4
	Self-Prompt	92.1	78.0	79.7	76.0	56.0	76.4
	IMR-TIP	**92.6**	**79.3**	**82.0**	**79.1**	**65.2**	**79.6**

Table 2: The overall results on the five datasets. We highlight the best results for each backbone. In our experiments, as we obtain 3 improved tool-based prompts through self-prompt (Section 3), we report their average results. SVAMP-H and GSM8K-H are short for SVAMP-Hard and GSM8K-Hard. We report average results of 3 runs.

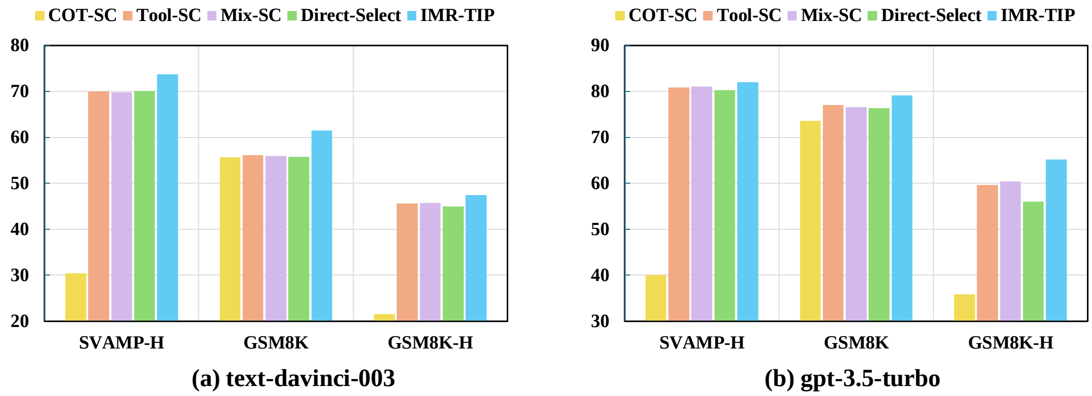

Figure 3: Ablation studies of different prompting methods based on `text-davinci-003` and `gpt-3.5-turbo`.

Models	SVAMP-H	GSM8K	GSM8K-H
IMR-TIP	**73.7**	**61.5**	**47.4**
w/ Step-I	70.4	56.3	44.5
w/ Step-II	72.4	58.7	45.6
w/ Step-I + II	72.9	59.4	46.5

Table 3: Ablation studies in our IMR-TIP. Here, we conduct results based on `text-davinci-003`.

integration for LLMs.

For ablations, we also compare ours with (1) *Self-consistency*: We build self-consistency baselines based on LLMs and their tool-augmented counterparts, named **COT-SC** and **Tool-SC**, separately. For a fair comparison, we sample reasoning paths 3 times by setting the temperature as 0.7. Then we also introduce **Mix-SC**, which votes the most consistent answer from the given solutions in TIP as the prediction. (2) *Direct Select*: Given a math problem, multiple solutions from LLMs and tool-augmented LLMs, we regard it as a multiple choice question answering task and require LLMs directly selecting one of them as the answer through COT. We show an example of it in Figure 1. See Appendix A for more details about their definitions.

Of note, although other works, such as PAL (Gao et al., 2023), employs tools like programming languages for math problem-solving, we do not compare our method with theirs in this paper to maintain fairness, as our tools differ.

3.3 Main Results

Table 2 shows the results of three backbone models with various prompting algorithms. Some key observations are as follows from the table:

LLMs still suffer in math calculations. It is evident that the performances of these two backbone LLMs with COT on SVAMP-Hard and GSM8K-Hard are notably lower compared to their performance on SVAMP and GSM8K (e.g., 77.1% vs.29.5% on SVAMP and SVAMP-Hard). This discrepancy suggests that when tasked with computations involving larger numerical values, these LLMs exhibit an increased susceptibility to calculation errors. Interestingly, GPT-3.5 shows better computation ability than GPT-3.

Self-prompt benefits the performances for tool-using. Across all datasets, it is discernible that the prompts subjected to self-prompt adjustments manifest enhanced performance relative to their original counterparts. This enhancement is attributed to the provision of more lucid and explicit tool usage instructions, alongside a heightened standardization of input-output formats.

IMR-TIP outperforms other baselines consistently, particularly on more challenging datasets involving complex computations. Firstly, the table highlights the tool-augmented method's superiority over the traditional COT approach on all datasets, attributed to enhanced computational abilities. Secondly, our approach yields additional enhancements across all datasets, notably pronounced in the more challenging SVAMP-Hard and GSM8K-Hard sets demanding higher computational prowess. For example, IMR-TIP based on GPT-3.5 could improve COT and Self-Prompt from 35.0%, 56.0% to 65.2% on GSM8K-Hard.

3.4 Ablation Study

In this part, we conduct ablation studies to verify the impact of (1) each key component in IMR-TIP; (2) self-prompt (**How well prompts from self-prompt perform specifically?** See Appendix B for more details.).

Key Components Given the sequential nature of our three-step IMP process (See Section 3), our ablation study comprises three sub-experiments in Table 3: 1) Step I only; 2) Step II only; 3) Steps I and II together. Table 3 shows that: 1. Step II holds paramount importance; 2. Steps I and III also exhibit significance; 3. The combined use of all three steps yields optimal performance.

4 Analysis

IMR-TIP vs. Self-consistency Given multiple solutions from LLMs and tool-augmented LLMs, two intuitive ways to derive the answer are: 1) *Self-Consistency*; 2) *Direct Select*. As aforementioned, we have three versions of self-consistency: COT-SC, Tool-SC and Mix-SC. Figure 3 shows the comparison between ours and these approaches on three datasets. We can observe that Tool-SC and Mix-SC perform comparably but they perform much better than COT-SC, just as tool-augmented LLMs surpass LLM-COT. Across two distinct backbone models, IMR-TIP consistently outperforms other methods on each dataset.

4.1 Why IMR-TIP works?

In pursuit of a comprehensive dissection of our IMR-TIP's underlying mechanics, we proffer a distributional analysis capturing the trajectory through which IMR-TIP reaches its definitive answers. Table 5 delineates the proportions in which IMR-TIP either selects answers from provided options or re-generates new answers on SVAMP-Hard and GSM8K-Hard datasets. The former is segmented into three categories of assessment by IMR-TIP: solely based on LLM's answer, based on both the LLM and the Tool-augmented methods, and exclusively within the Tool-augmented method. Predominantly, IMR-TIP favors answers from Tool-augmented LLMs that are derived from provided solutions. Notably, IMR-TIP's capacity for generating new answers through observation and cross-checking significantly enhances its performance relative to self-consistency. This unique capability distinguishes IMR-TIP from other methods.

We remain curious about the performance of IMR-TIP when not all the given options are necessarily correct. To verify this, we conduct in-depth analysis of the GSM8K-Hard dataset in Table 6 shows that our model incorrectly answers 2.7% of cases when all provided answer choices are correct. Conversely, it correctly answers 5.2% of cases when all choices are incorrect. These outcomes highlight the challenges in enabling a model to self-verify and self-correct. Despite some errors, **our method strengthens LLM's ability to enhance its understanding and correct mistakes through interleaving reasoning steps**, which is a central aspect of our approach.

Case Study Table 4 shows the an example from IMR-TIP and self-consistency on the GSM8K dataset. In this instance, only option (D) stands correct, with option (A) exhibiting a computational error, and both options (B) and (C) featuring accurate algebraic expressions albeit flawed numerical substitutions. Within this context, Mix-SC would favor the most consistently occurring answer, leading to the selection of 44 as the predicted answer, despite its inaccuracy. While our approach also encounters numerical substitution errors in [Tho-2], meticulous scrutiny of both the answer and reasoning steps reveals the preceding mistake, affirming the correctness of option (D). This case underscores another inherent advantage of IMR-TIP over self-consistency: the capacity to derive an accurate answer through thoughtful reasoning, rather than a

Question: Sadie slept 8 hours on Monday. For next two days, she slept 2 hours less, each, because she had to complete some assignments. If the rest of the week she slept 1 hour more than those two days, how many hours did she sleep in total throughout the week?
(A): Sadie slept 8 hours on Monday. For the next two days, she slept 2 hours less, each, so she slept 6 hours each day. If the rest of the week she slept 1 hour more than those two days, she slept 7 hours each day. So the total amount of hours she slept throughout the week is 8+6+6+7+7+7+7=42. So the answer is 42.
(B): The mathematical equation to solve the problem is: total hours slept = (Monday's hours + (2 days * (hours - 2)) + (4 days * (hours + 1))). Substituting the given values, we get: total hours slept = (8 + (2 * (6 - 2)) + (4 * (6 + 1))). Therefore, Answer = Calculator[8 + (2 * (6 - 2)) + (4 * (6 + 1))] = 44.
(C): [Algebraic Equation]: Total hours slept = (Monday's hours + (2 days * (Monday's hours - 2 hours)) + (4 days * (Monday's hours - 2 hours + 1 hour)) [Numeric Equation]: Total hours slept = (8 + (2 * (8 - 2)) + (4 * (8 - 2))) [Answer]: Calculator[(8 + (2 * (8 - 2)) + (4 * (8 - 2)))] = 44.
(D): [Algebraic Equation]: total hours slept = hours slept on Monday + (hours slept on Tuesday + hours slept on Wednesday + hours slept on Thursday + hours slept on Friday + hours slept on Saturday + hours slept on Sunday) [Numeric Equation]: total hours slept = 8 + (6 + 6 + 7 + 7 + 7 + 7) [Answer]: Calculator[8 + (6 + 6 + 7 + 7 + 7 + 7)] = 48.

IMR-TIP:
Tho-1: To validate different solutions, I first need to analyze the math problem and give my reasoning logic.
Act-1: ***Analyze[problem]***
Tho-2: I need to find the total amount of hours Sadie slept throughout the week given the information that she slept 8 hours on Monday, 2 hours less for the next two days, and 1 hour more for the rest of the week. Thus, the total hours = 8 + (2 * (8 - 2)) + (4 * (8 - 2)). To ensure the calculation is correct, I need to use Calculator to calculate the answer.
Act-2: Calculator[8 + (2 * (8 - 2)) + (4 * (8 - 2))]
Tho-3: With the answer hint 44, I can first compare their answers to make an initial assessment of their accuracy.
Act-3: ***Compare[answers]***
Tho-4: I observe that (A) directly gives the answer while (B), (C) and (D) use the Calculator to compute the answer. The answers of them are 42, 44, 44, 48. Half of them align with the answer hint 44, I can preliminarily conclude that (B), (C) are correct. But there are still two different answers, I cannot determine which answer is definitely correct. Thus, I can compare their reasoning steps and equations with mine to further verify their correctness.
Act-4: ***Compare[reasoning]***
Tho-5: (A) and (D) follow similar reasoning paths and their mathematical equations are the same, but their answers are different. Due to (D) using the calculator, (A) must have mistakes in calculation. (B) and (C) give similar algebraic equations, but their substitutions are incorrect. (B) states that Sadie slept 6 hours each day for the next two days, which is not accurate. (C) is also incorrect. I may also make a mistake in [Tho-2] because my numerical equation is the same as (C). I need to review [Tho-2] again to validate the accuracy of my previous reasoning and answer hint.
Act-5: ***Rethink[Tho-2]***.
Tho-6: Based on my findings, the algebraic equation in [Tho-2] is correct, but the substitution is incorrect. The correct total hours = 8 + (2 * (8 - 2)) + (4 * (8 - 2 + 1)) rather than 8 + (2 * (8 - 2)) + (4 * (8 - 2)).
Act-6: Calculator[8 + (2 * (8 - 2)) + (4 * (8 - 2 + 1))]
Tho-7: After revising the substitution in [Tho-2], I think the final answer is 48 and (D) is correct.
Ans: **48**

Mix-SC: 44.

Table 4: Examples of generated solutions by IMR-TIP and self-consistency on GSM8K dataset.

	Answer Type	SVAMP-H	GSM8K-H
Select	Only LLM	4.5%	6.3%
	LLM and Tool	13.4%	18.5%
	Only Tool	79.0%	67.1%
Re-generate new answers		3.1%	8.1%

Table 5: Proportion of each answer type in our IMR-TIP. Here, we conduct results based on `gpt-3.5-turbo`.

Method	Answer options have the right answer?			
	GSM8K-H (%)	Yes	No	Total
IMR-TIP	Wrong	**2.7%**	31.8%	34.5%
	Right	60.3%	**5.2%**	65.5%
	Total	63.0%	37.0%	100%

Table 6: Ours performance when provided with answer choices that are either correct or incorrect in GSM8K-Hard dataset. The experimental LLMs is `gpt-3.5-turbo`.

simplistic reliance on majority voting for consistency. We present more cases in Appendix C.

5 Conclusion

This work introduces IMR-TIP, a framework designed to capitalize on the strengths of both conventional and tool-augmented LLMs, enhancing their mathematical capabilities. IMR-TIP employs self-prompt to refine its initial prompts. Additionally, it utilizes tool-augmented interleaf prompting (TIP) to reach the correct answer. This process involves interleaved *Action* and *Thought*, which encompasses analyzing the problem, observing, cross-verifying different solutions, and even rethinking previous answer hints. Through rigorous experimentation and comprehensive analysis of math reasoning tasks, IMR-TIP demonstrates a substantial enhancement over established baselines.

Limitations

In our study on enhancing the mathematical capabilities of Large Language Models (LLMs) through prompt engineering, we acknowledge two primary limitations: the potential for inaccuracies and the concern over prompt length. Despite the method's promising results, it is susceptible to errors in specific scenarios, which may affect the model's overall reliability. Moreover, the use of extended prompts, while instrumental in our approach, introduces practical constraints related to processing time and model usability. Recognizing these challenges, future research will focus on developing training techniques aimed at improving the LLM's self-revision abilities and optimizing prompt length. This direction is expected to mitigate current limitations and further the application of LLMs in complex problem-solving tasks.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020a. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020b. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Nuo Chen, Hongguang Li, Yinan Bao, Junqing He, Xinshi Lin, Qi Yang, Jianfeng Liu, Ruyi Gan, Jiaxing Zhang, Baoyuan Wang, et al. 2023a. Orca: A few-shot benchmark for chinese conversational machine reading comprehension. *arXiv preprint arXiv:2302.13619*.

Nuo Chen, Zinan Zheng, Ning Wu, Linjun Shou, Ming Gong, Yangqiu Song, Dongmei Zhang, and Jia Li. 2023b. Breaking language barriers in multilingual mathematical reasoning: Insights and observations. *arXiv preprint arXiv:2310.20246*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

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. *CoRR*, abs/2110.14168.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. In *ACL (industry)*, pages 37–42. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333.

Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2022. A survey of deep learning for mathematical reasoning. *arXiv preprint arXiv:2212.10535*.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. *arXiv preprint arXiv:2301.13379*.

OpenAI. 2023. Gpt-4 technical report.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. 2023. Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*.

Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Jing Qian, Hong Wang, Zekun Li, Shiyang Li, and Xifeng Yan. 2022. Limitations of language models in arithmetic and symbolic induction. *arXiv preprint arXiv:2208.05051*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Yixuan Weng, Minjun Zhu, Shizhu He, Kang Liu, and Jun Zhao. 2022. Large language models are reasoners with self-verification. *arXiv preprint arXiv:2212.09561*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Chenyu You, Nuo Chen, Fenglin Liu, Shen Ge, Xian Wu, and Yuexian Zou. 2022. End-to-end spoken conversational question answering: Task, dataset and model. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1219–1232, Seattle, United States. Association for Computational Linguistics.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. 2023. How well do large language models perform in arithmetic tasks? *arXiv preprint arXiv:2304.02015*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Related Work

Math Reasoning with LLMs In recent times, large language models (LLMs) have demonstrated remarkable abilities in handling complex reasoning tasks (Scao et al., 2022; Cobbe et al., 2021; Zhou et al., 2022; Weng et al., 2022; Chen et al., 2023b; Gou et al., 2023). Rather than providing direct final answers as outputs, prior research has shown that by employing diverse prompting methods such as Chain-of-Thought (COT) prompting (Wei et al., 2022), LLMs can be guided through step-by-step reasoning, resulting in notably improved performance across a wide array of reasoning tasks. Imani et al. (2023) propose to generate multiple algebraic expressions or Python functions to solve the same math problem, aiming to explore different potential solutions. Li et al. (2023) introduce a step-aware verifier to check the reasoning steps in COT, improving the reasoning capabilities. Self-Consistency (Wang et al., 2022) is another effective work that combines different solutions and gets a final answer by aggregating to retrieve the most consistent answer. Among them, self-consistency bears resemblance to our proposed TIP, but our main distinction lies in the following: Given several solutions, our TIP can re-generate a new answer after cross-validating these solutions. In contrast, self-consistency can only select the most consistent answer from the existing ones.

Tool-Augmented LLMs Currently, researchers have undertaken a wide array of studies aimed at enriching the step-by-step reasoning process. These approaches include investigating the utilization of external tools (Parisi et al., 2022; Schick et al., 2023; Yao et al., 2022), like program interpreters (Lyu et al., 2023; Chen et al., 2022) and the calculator (Schick et al., 2023), training and utilizing external reasoning modules. For example, ReAct (Yao et al., 2022) proposes an interleaved framework with utilizing an external search engine to solve multi-hop question answering tasks. More recently, Toolformer (Schick et al., 2023) introduces a pipeline for training LLMs that can call tools during training and inference. Parallel to these works, our work can be seen as a preliminary ex-

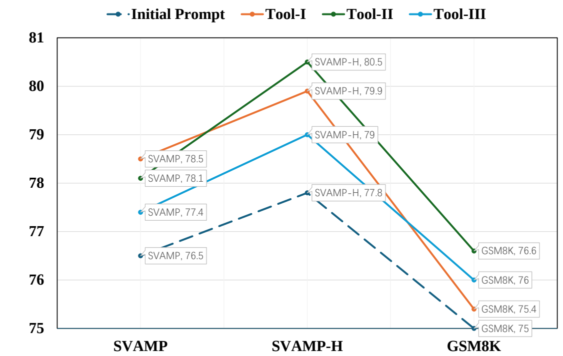

Figure 4: Model performances of different prompts. Here, we use the gpt-3.5-turbo as the backbone LLM.

ploration of *how to better utilize tools*, which involves a fusion of diverse solutions of LLMs and tool-augmented LLMs, culminating in the attainment of the final answer through tool-augmented interleaf prompting.

B Appendix A: Prompts

Table 7 presents COT prompts in our experiments. Table 8 shows three tool-based prompts obtained from our proposed self-prompt. It is worth noting that in most cases, the self-prompt does not alter the examples within the prompt. Its primary focus lies in modifying instructions related to tool usage, input-output formats, and tool definitions. All of them contain only 2 examples.

Table 9 shows the prompts of TIP in our experiments. Practically, we append "——" as the stopping symbols after each *Action* to achieve interleaving.

B.1 Definition of Self-consistency and Direct-Select

- COT-SC: Utilizing COT prompts in Table 1 for math reasoning and sampling 3 times.
- COT-SC: Utilizing tool-based prompts in Table 2 for math reasoning and sampling 3 times. Then we report their average results.
- Mix-SC: Selecting the most consistent answer from given options (A, B, C, D) in TIP.
- Directly Select: Select the answer from the given options in TIP through COT. Prompts are shown in Table 1.

C Appendix B: Self-Prompt

In this component, we conduct ablation studies to answer the question: How well do prompts from self-prompt perform? Figure 4 depicts a comparison between the initial tool-based prompt and the three prompts modified through self-prompt. It is evident across all three datasets that the prompts refined by self-prompt exhibit improved performance compared to the original initial prompts. This enhancement can be attributed to the provision of clearer, more explicit tool usage instructions and more standardized input-output formats. Overall, our self-prompt follows the road "LLMs know themselves better." The most simple implementation of self-prompt is to interact with ChatGPT or GPT-4 in OpenAI' website[2].

D Appendix C: More Cases

In this section, we present typical cases when IMR-TIP solves math reasoning tasks.

Table 10 presents the most straightforward scenario, wherein the provided solution corresponds with the answer generated by IMR-TIP in [Tho-2]. In this instance, the LLM detects the uniformity of all five answers upon comparison, resulting in a notable elevation of confidence and the subsequent direct issuance of the final answer.

Table 11 also provides an example wherein IMR-TIP makes an error. Instead of further comparing the reasoning steps of different solutions after analyzing the answers, the model directly presents a conclusion, asserting the correctness of its approach.

Table 12 introduces another intriguing case. In this instance, IMR-TIP attains the correct answer for [Tho-2]. However, upon reevaluating the preceding reasoning hints and formulas for [Tho-6], it identifies a numerical substitution error within [Tho-2]. Despite this recognition, IMR-TIP proceeds to output the identical formula as the final answer. Instances in which LLMs exhibit self-contradictory reasoning yet still yield the correct ultimate answer in IMR-TIP, as exemplified here, are referred to as "FALSE POSITIVE."

[2] chat.openai.com

COT prompts
Your task is to answer the following math questions. Question: julia played tag with 18 kids on monday . she played tag with 10 kids on Tuesday. how many more kids did she play with on monday than on Tuesday? Answer: Let's think step by step. julia playsed tag with 18 kids on monday and 10 kids tuesday, separately. So the amount of kids that she played with on monday than on tuesday is 18-10=8. So the answer is 8. Question: Jack had 9 action figures and 10 books on a shelf in his room. later he added 7 more action figures to the shelf. how many more action figures than books were on his shelf? Answer: Let's think step by step. The amount of action figures that Jack had is 9+7=16. And Jack had 10 books. So the amount of action figures than books on his shelf is 16-10=6. So the answer is 7.
Initial tool-based prompts
Your task is to solve the following middle-school arithmetic problems by using the Calculator. A calculator is a system used for performing mathematical calculations, ranging from basic arithmetic to more complex functions. You can do so by writing a mathematical equation and generating the answer format starting with "Answer = Calculator[expression]", where "expression" is the expression to be computed. Below are examples: Question: julia played tag with 18 kids on Monday. she played tag with 10 kids on Tuesday. how many more kids did she play with on monday than on Tuesday? Thought: The mathematical equation to solve the problem is: number of additional kids played with = number of kids played with on Monday - number of kids played with on Tuesday. Substituting the given values, we get: number of additional kids played with = 18 - 10. Therefore, Answer = Calculator[18 - 10] Question: Jack had 9 action figures and 10 books on a shelf in his room. later he added 7 more action figures to the shelf. how many more action figures than books were on his shelf? Thought: The mathematical equation to solve the problem is: difference in number of action figures and books = (number of action figures initially + number of action figures added) - number of books. Substituting the given values, we get: difference in number of action figures and books = (9 + 7) - 10. Therefore, Answer = Calculator[(9 + 7)- 10]
Direct-select prompts
There is a multiple choice question answering task. You are required to select one of the options to answer the problem. Let's think step by step.

Table 7: LLMs-COT, initial tool-based prompts and direct-select prompts in our experiments.

Prompt of Tool-I for math reasoning tasks.

Your task is to solve the following middle-school arithmetic problems using a calculator. Follow the steps provided to write out the algebraic and numeric equations, and use a calculator to find the answers. You can generate the answer format starting with “Answer = Calculator[expression]", where “expression" is the expression to be computed.
Example Problems:
Question: Julia played tag with 18 kids on Monday and 10 kids on Tuesday. How many more kids did she play with on Monday than on Tuesday?
Thought:
Algebraic Equation: Additional kids played with = Kids played with on Monday - Kids played with on Tuesday.
Numeric Equation: Additional kids played with = 18 - 10
Answer: Calculate Calculator[18 - 10]

Question : Jack had 9 action figures and 10 books on a shelf. Later, he added 7 more action figures to the shelf. How many more action figures than books were on his shelf?
Thought:
Algebraic Equation: Difference in action figures and books = (Initial action figures + Added action figures) - Number of books.
Numeric Equation: Difference in action figures and books = (9 + 7) - 10
Answer: Calculate Calculator[(9 + 7) - 10]
Remember to enter the appropriate equation into your calculator to find the answer. You can also explain how you would verify your solution.

Prompt of Tool-II for math reasoning tasks.

Your task is to solve the following middle-school arithmetic problems by using the Calculator. A calculator is a system used for performing mathematical calculations, ranging from basic arithmetic to more complex functions.

You can do so by writing out Algebraic Equation, Numeric Equation, Answer steps. In the Answer step, you should generate the output format with “Calculator[expression]", where “expression" is the expression to be computed.
Below are examples:
Question: julia played tag with 18 kids on monday. she played tag with 10 kids on Tuesday. how many more kids did she play with on monday than on Tuesday?
Thought: [Algebraic Equation]: number of additional kids played with = number of kids played with on Monday - number of kids played with on Tuesday.
[Numeric Equation]: number of additional kids played with = 18 - 10.
[Answer]: Calculator[18 - 10]

Question: Jack had 9 action figures and 10 books on a shelf in his room. later he added 7 more action figures to the shelf. how many more action figures than books were on his shelf?
Thought: [Algebraic Equation]: difference in number of action figures and books = (number of action figures initially + number of action figures added) - number of books.
[Numeric Equation]: difference in number of action figures and books = (9 + 7) - 10
[Answer]: Calculator[(9 + 7)- 10]

Prompt of Tool-III for math reasoning tasks.

Your task is to solve the following middle-school arithmetic problems by using the Calculator. A Calculator is a system used for performing mathematical calculations, ranging from basic arithmetic to more complex functions.

You can do so by writing out Algebraic Equation, Numeric Equation, and Answer steps:
(1) [Algebraic Equation], which directly builds an Algebraic Equation by using variables from the question to generate the solution.
(2) [Numeric Equation], which plugs in values from the question to transform the above Algebraic Equation into its numeric form.
(3) [Answer], which uses Calculator to calculate the above numeric equation, but not directly to give the final answer. You can generate the answer format with “Calculator[expression]", where “expression" is the expression to be computed.

Below are examples:
Question: julia played tag with 18 kids on monday. she played tag with 10 kids on tuesday. how many more kids did she play with on monday than on Tuesday?
Thought: [Algebraic Equation]: number of additional kids played with = number of kids played with on Monday - number of kids played with on Tuesday.
[Numeric Equation]: number of additional kids played with = 18 - 10.
[Answer]: Calculator[18 - 10]

Question: Jack had 9 action figures and 10 books on a shelf in his room. later he added 7 more action figures to the shelf. how many more action figures than books were on his shelf?
Thought: [Algebraic Equation]: difference in the number of figures and books = (number of figures initially + number of figures added) - number of books.
[Numeric Equation]: difference in the number of figures and books = (9 + 7) - 10
[Answer]: Calculator[(9 + 7)- 10]
You should keep in mind:
(1) you can not image or output the result of "Calculator[expression]".
(2) When generating a mathematical formula, do not include any units of measurement.

Table 8: Three tool-based prompts from Self-Prompt.

Your task is to validate the several options to the same math problem and select the correct ones. You should make your own reflections, revise your own thoughts, give the final answers based on the previous findings at the end. You can use the Calculator to help you calculate the numeric equations and verify the solutions.
A Calculator is a system used for performing mathematical calculations, ranging from basic arithmetic to more complex functions.
To do this, you can writing out interleaving Thought, Action and Answer steps. Notice that, you need to end with Answer when solving the task.
Below are examples:
Question: A robe takes 2287720 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?
(A): A robe ...... 2287720 + (2287720/2) = 3,431,280 bolts.
(B): Based on the statement of the question ...... total number of bolts = Calculator[2287720 + (2287720/2)] = 3431580.
(C): [Algebraic Equation]: total number of bolts = number of blue bolts + number of white bolts...... Calculator[2287720 + (2287720 / 2)] = 3431580.
(D): [Algebraic Equation]: total number of bolts ......Calculator[2287720 + (2287720/2)] = 3431580.
Thoughts: [Thought-1]: To validate different solutions, I first need to analyze the math problem and give my reasoning logic.
[Action-1]: Analyze[problem]
[Thought-2]: First, the number of bolts of while fiber is half of the number of bolts of blue fiber. Total number of bolts = Number of bolts of blue fiber + Number of bolts of white fiber. Thus, total number of bolts = 2287720 + 2287720/2. To ensure the calculation is correct, I should use Caculator to calculate the answer.
[Action-2]: Calculator[2287720 + 2287720/2]
[Calculated Result]: 3431580
[Thought-3]: With the answer hint 3431580, I can first compare the answers of different options to make an initial assessment of their accuracy.
[Action-3]: Compare[answers]
[Thought-4]: The answer of option (A) is 3431280 that contradicts the answer hint 3431580, while options (B), (C), and (D) align with the answer hint. Therefore, I can preliminarily conclude that (A) is incorrect, and (B), (C), and (D) are correct. Since most of the answer options are consistent with my previous answer hint, I believe the current answer hint is likely correct. I can compare their reasoning steps and equations with mine to further verify my judgment and investigate why option (A) yielded an incorrect answer.
[Action-4]: Compare[reasoning]
[Thought-5]: Option (A) follows a similar reasoning process and formula as mine in [Thought-2], suggesting an error in the final calculation. Options (B), (C), and (D) provide equations that align with my earlier reasoning. Based on my findings, upon comparing the answers and reasoning processes, options (B), (C), and (D), along with the answer hint, are all correct.
[Answer]: Calculator[2287720 + 2287720/2]

Question: Trent is 5 years older than Jane, and Jane is 3 years younger than Quinn. If Quinn is 2538570, how old is Trent?
(A): Quinn is 2538570...... So the answer is Trent is 2538572.
(B): The mathematical equation to solve the problem is: ......Answer = Calculator[2538570 + (3 - 2538570) + 5] = 8.
(C): [Algebraic Equation]: Trent's age = Quinn's age ......2538578.
(D): [Algebraic Equation]: ...... Calculator[2538570 + 5 + 3] = 2538578.
Thoughts: [Thought-1]: To validate different solutions, I first need to analyze the math problem and give my reasoning logic.
[Action-1]: Analyze[problem]
[Thought-2]: I need to find Trent's age given the relationships between Trent, Jane, and Quinn's ages. Jane is 3 years younger than Quinn and 5 year younger than Trent. If Quinn is 2538570, Trent's age is 2538570 -3 + 5. To ensure the calculation is correct, I need to use Caculator to calculate the answer.
[Action-2]: Calculator[2538570 -3 + 5]
[Calculated Result]: 2538572
[Thought-3]: With the answer hint 2538572, I can frist compare the answers of different options to make an initial assessment of their accuracy.
[Action-3]: Compare[answers]
[Thought-4]: The answer of option (A) is 2538572 that align with the answer hint 2538572. However options (B)'s answer is 8, the answers of (C) and (D) are both 2538578 which all contradict the answer hint. Therefore, I can preliminarily conclude that (A) is correct, and (B), (C), and (D) are incorrect. But there are three different answers and only one aligns with the answer hint, I cannot determine which answer is definitely correct. Thus, I can compare their reasoning steps and equations with mine to further verify their correctness.
[Action-4]: Compare[reasoning]
[Thought-5]: (A) follows a similar reasoning process and formula as mine in [Thought-2]. (B) is correct in terms of the mathematical equation used, but the substitution is incorrect. It states that Jane's age is equal to 3. (C) and (D) present that Trent's age = Quinn's age + 8 (5+3), meaning that Trent is 5 years older that Jane, and Jane is 3 year older than Quinn which contradict the information given in the question. The correct algebraic equation would be to subtract 3 from Quinn's age to determine Jane's age and then add 5 to Jane's age to find Trent's age. Based on my findings, I think (B), (C) and (D) are wrong, the answer hint and (A) are correct with no need improvements. I don't need to review my previous answer hint.
[Answer]: Calculator[2538570 -3 + 5]

Question: Carla is downloading a 200 GB file. Normally she can download 2 GB/minute, but 40 percent of the way through the download, Windows forces a restart to install updates, which takes 20 minutes. Then Carla has to restart the download from the beginning. How load does it take to download the file?
(A): It will take her 200 minutes to download the file.
(B): The mathematical equation ...... Therefore, Answer = Calculator[(0.4 * (200/2)) + 20] = 60.
(C): [Algebraic Equation]: ...... Calculator[(200/2) + (200/2 * (1 - 0.4))/2] = 130.
(D): [Algebraic Equation]: total time ......[Answer]: Calculator[(200/2 + (200 - (200*0.4))/2) / 2] = 180.
Thoughts: [Thought-1]: To validate different solutions, I first need to analyze the math problem and give my reasoning logic.
[Action-1]: Analyze[problem]
[Thought-2]: First, I should list the equation to calculate the time it took Carla to download the data before the Windows restart: time before restart = Downloaded data/Download speed. And time after restart = File size/Donwload speed. Thus, the total time = time before restart + Windows update + time after restart. Thus, total time = 200 *0.4/2/2 + 20 + 200/2. To ensure the calculation is correct, I need to use Calculator to calculate the answer.
[Action-2]: Calculator[200 * 0.4/2/2 + 20 + 200/2]
[Calculated Result]: 140
[Thought-3]: With the answer hint 140, I can first compare the answers of different options to make an initial assessment of their accuracy.
[Action-3]: Compare[answers]
[Thought-4]: The answers of these options are 200, 60, 130 and 180, separately, that are different with the answer hint 140. Thus, I can not preliminarily conclude which one is correct. I need to further compare their reasoning steps and equations with mine to further verify their correctness.
[Action-4]: Compare[reasoning]
[Thought-5]: (B), (C), and (D) have flawed algebraic equations as they omit the inclusion of Windows update time mentioned in the reasoning hint. (A) lacks reasoning steps, making its correctness uncertain. Because none options have the same answer as mine, I need to review [Thought-2] again to validate the accuracy of my previous reasoning and answer hint.
[Action-5]: Rethink[Thought-2]
[Thought-6]: The algebraic equation in [Thought-2] is correct, but the substitution is incorrect. The correct total time = 200 *0.4/2 + 20 + 200/2 rather than 200 *0.4/2/2 + 20 + 200/2. So based on my findings, the answer is Calculator[200 * 0.4/2 + 20 + 200/2].
[Answer]: Calculator[200 * 0.4/2 + 20 + 200/2]

Keep in mind the following requirements:
(1) You must output the correct equation as the final answer in [Answer].
(2) If answers of all options are the same as the answer hint, you can directly end the task and output the correct equation in Answer.
(3) If you decide to use Calculator in Action, generate the format with "Calculator[expression]", where "expression" is the expression to be computed.
(4) Do not include any units of measurement in "Calculator[expression]".
(5) You can not image the results of "Calculator[expression]".

Table 9: Prompts of TIP. In this table, we omit details of each option.

Question: There were some birds sitting on the fence. 725067 more birds came to join them. if there are a total of 544650 birds on the fence now how many birds had been sitting on the fence at the start?
(A): The total number of birds on the fence now is 544650. And 725067 more birds came to join them. So the total number of birds at the start is 544650 - 725067 = -180417.
(B): Answer = Calculator[544650 - 725067] = -180417
(C): [Algebraic Equation]: number of birds at the start = total number of birds on the fence now - number of birds that came to join them. [Numeric Equation]: number of birds at the start = 544650 - 725067 [Answer]: Calculator[544650 - 725067] = -180417
(D): [Algebraic Equation]: number of birds at the start = total number of birds on the fence now - number of birds that came to join them. [Numeric Equation]: number of birds at the start = 544650 - 725067 [Answer]: Calculator[544650 - 725067] = -180417

IMR-TIP:
Tho-1: To validate different solutions, I first need to analyze the math problem and give my reasoning logic.
Act-1: Analyze[problem]
Tho-2: The problem states that there were some birds sitting on the fence, and 725067 more birds came to join them. The total number of birds on the fence now is 544650. To find the number of birds that had been sitting on the fence at the start, we can subtract the number of birds that came to join from the total number of birds now. To ensure the calculation is correct, I need to use the Calculator to calculate the answer.
Act-2: Calculator[544650 - 725067]
Tho-3: With the answer hint -180417, I can first compare the answers of different options to make an initial assessment of their accuracy.
Act-3: Compare[answers]
Tho-4: The answers of options (A), (B), (C), and (D) are all -180417, which align with the answer hint -180417. Therefore, I can conclude that all options are correct. No more action are needed.
Ans: **-180417.0**

Ground-Truth Anaswer: **-180417.0**

Table 10: The MOST SIMPLE interleaved example of generated solutions by IMR-TIP on SVAMP-Hard dataset.

Question: julia played tag with 829557 kids on monday and 853729 kids on tuesday. she played cards wtih 913524 kids on Wednesday. how many kids did she play tag with altogether?
(A): Julia played tag with 829,557 kids on Monday and 853,729 kids on Tuesday. To find out how many kids she played tag with altogether, we need to add these two numbers together. 829,557 + 853,729 = 1,683,286 So, Julia played tag with a total of 1,683,286 kids altogether.
(B): Answer = Calculator[829557 + 853729] = 1683286
(C): [Algebraic Equation]: total number of kids played tag with = number of kids played tag with on Monday + number of kids played tag with on Tuesday + number of kids played tag with on Wednesday. [Numeric Equation]: total number of kids played tag with = 829557 + 853729 + 913524 [Answer]: Calculator[829557 + 853729 + 913524] = 2596810
(D): [Algebraic Equation]: total number of kids played tag with = number of kids played tag with on Monday + number of kids played tag with on Tuesday + number of kids played tag with on Wednesday. [Numeric Equation]: total number of kids played tag with = 829557 + 853729 + 913524 [Answer]: Calculator[829557 + 853729 + 913524] = 2596810

IMR-TIP:
Tho-1: To validate different solutions, I first need to analyze the math problem and give my reasoning logic.
Act-1: Analyze[problem]
Tho-2: Julia played tag with a certain number of kids on Monday, Tuesday, and Wednesday. To find out how many kids she played tag with altogether, I need to add the number of kids she played tag with on each day. The equation to calculate the total number of kids played tag with is: total number of kids played tag with = number of kids played tag with on Monday + number of kids played tag with on Tuesday + number of kids played tag with on Wednesday. To ensure the calculation is correct, I need to use the Calculator to calculate the answer.
Act-2: Calculator[829557 + 853729 + 913524]
Tho-3: With the answer hint 2596810, I can first compare the answers of different options to make an initial assessment of their accuracy.
Act-3: Compare[answers]
Tho-4: The answers of options (A), (B), (C), and (D) are 1,683,286, 1,683,286, 2,596,810, and 2,596,810, respectively. The answer hint is 2,596,810. Therefore, options (A) and (B) are incorrect, while options (C) and (D) are correct. Since options (C) and (D) have the same answer and reasoning process, I can conclude that the correct answer is Calculator[829557 + 853729 + 913524].
Ans: **2596810**

Ground-Truth Anaswer: **1683286.0**

Table 11: The typical NEGATIVE interleaved example of generated solutions by IMR-TIP on SVAMP-Hard dataset.

Question: Sadie slept 8 hours on Monday. For next two days, she slept 2 hours less, each, because she had to complete some assignments. If the rest of the week she slept 1 hour more than those two days, how many hours did she sleep in total throughout the week?
(A): Wendi feeds each of her chickens three cups of mixed chicken feed, containing seeds, mealworms and vegetables to help keep them healthy. So the amount of cups of feed that Wendi needs to give her chickens in the final meal of the day is 20x3=60. So the answer is 60.
(B): The mathematical equation to solve the problem is: total cups of feed = (number of chickens * cups of feed per meal) - (cups of feed given in morning + cups of feed given in afternoon). Substituting the given values, we get: total cups of feed = (20 * 3) - (6887483 + 25). Therefore, Answer = Calculator[(20 * 3) - (6887483 + 25)] = -6887448.
(C): [Algebraic Equation]: number of cups of feed for the final meal = (number of chickens * cups of feed per chicken) - (cups of feed given in the morning + cups of feed given in the afternoon). [Numeric Equation]: number of cups of feed for the final meal = (20 * 3) - (6887483 + 25) [Answer]: Calculator[(20 * 3) - (6887483 + 25)] =-6887448
(D): [Algebraic Equation]: total cups of feed for the final meal = (number of chickens * cups of feed per chicken) - (cups of feed given in the morning + cups of feed given in the afternoon) [Numeric Equation]: total cups of feed for the final meal = (20 * 3) - (6887483 + 25) [Answer]: Calculator[(20 * 3) - (6887483 + 25)] = -6887448

IMR-TIP:
Tho-1: To validate different solutions, I first need to analyze the math problem and give my reasoning logic.
Act-1: Analyze[problem]
Tho-2: Wendi needs to give her chickens three cups of feed in three separate meals. She has already given 6887483 cups of feed in the morning and 25 cups of feed in the afternoon. Thus, the number of cups of feed that Wendi needs to give her chickens in the final meal of the day is (Number of chickens * Cups of feed per chicken) - (Cups of feed given in the morning + Cups of feed given in the afternoon). To ensure the calculation is correct, I need to use Calculator to calculate the answer.
Act-2: **Calculator[(20 * 3) - (6887483 + 25)]**
Tho-3: With the answer hint -6887448, I can first compare the answers of different options to make an initial assessment of their accuracy.
Act-3: Compare[answers]
Tho-4: **The answers of these options are 60, -6887448, -6887448 and -6887448, separately, which are different with the answer hint -6887448.** Thus, I can not preliminarily conclude which one is correct. I need to further compare their reasoning steps and equations with mine to further verify their correctness.
Act-4: Compare[reasoning]
Tho-5: Option (A) follows a similar reasoning process as mine in [Tho-2], but the equation is incorrect. Options (B), (C), and (D) provide equations that align with my earlier reasoning. However, the substitution is incorrect as it does not include the cups of feed given in the morning and afternoon. Based on my findings, upon comparing the answers and reasoning processes, options (A), (B), (C), and (D) are all incorrect. I need to review [Tho-2] again to validate the accuracy of my previous reasoning and answer hint.
Act-5: Rethink[Tho-2].
Tho-6: The algebraic equation in [Tho-2] is correct, but the substitution is incorrect. The correct total cups of feed = (Number of chickens * Cups of feed per chicken) - (Cups of feed given in the morning + Cups of feed given in the afternoon). **So based on my findings, the answer is Calculator[(20 * 3) - (6887483 + 25)]**.
Act-6: Calculator[(20 * 3) - (6887483 + 25)]
Ans: **-6887448**

Ground-Truth Anaswer: **-6887448**

Table 12: FALSE POSITIVE Examples of generated solutions by IMR-TIP on GSM8K-Hard dataset.