ToRA: A Tool-Integrated Reasoning Agent for Mathematical Problem Solving

Zhibin Gou[1,2*], **Zhihong Shao**[1,2*], **Yeyun Gong**[2†], **Yelong Shen**[2]
Yujiu Yang[1†], **Minlie Huang**[1†], **Nan Duan**[2], **Weizhu Chen**[2]
[1]Tsinghua University [2]Microsoft
`{gzb22,szh19}@mails.tsinghua.edu.cn`
`{yegong,yeshe,nanduan,wzchen}@microsoft.com`

Abstract

Large language models have made significant progress in various language tasks, yet they still struggle with complex mathematics. In this paper, we propose ToRA, a series of Tool-integrated Reasoning Agents designed to solve challenging mathematical problems by seamlessly integrating natural language reasoning with the utilization of external tools (e.g., computation libraries and symbolic solvers), thereby amalgamating the analytical prowess of language and the computational efficiency of tools. To train ToRA, we curate interactive tool-use trajectories on mathematical datasets, apply imitation learning on the annotations, and propose output space shaping to further refine models' reasoning behavior. As a result, ToRA models significantly outperform open-source models on 10 mathematical reasoning datasets across all scales with 13%-19% absolute improvements on average. Notably, ToRA-7B reaches 44.6% on the competition-level dataset MATH, surpassing the best open-source model WizardMath-70B by 22% absolute. ToRA-Code-34B is also the first open-source model that achieves an accuracy exceeding 50% on MATH, which significantly outperforms GPT-4's CoT result, and is competitive with GPT-4 solving problems with programs. Additionally, we conduct a comprehensive analysis of the benefits and remaining challenges of tool interaction for mathematical reasoning, providing valuable insights for future research[1].

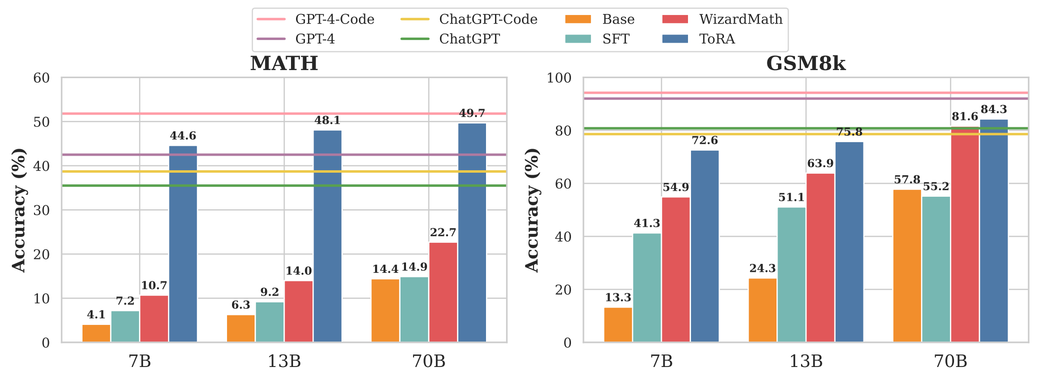
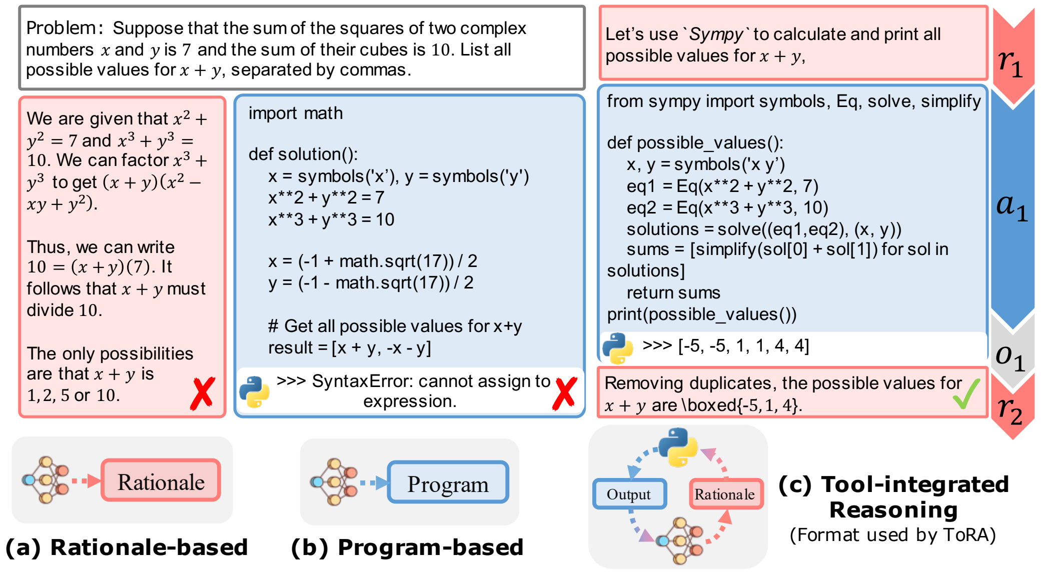


Figure 1: Comparing ToRA with baselines on LLaMA-2 base models from 7B to 70B. ToRA models exhibit remarkable improvements over previous state-of-the-art approaches across all scales. In particular, ToRA-70B notably outperforms GPT-4's CoT result on MATH and attains comparable results to GPT-4 solving problems with code.

[*]Equal contribution. See Contributions section for details. Work done during an internship at Microsoft.
[†]Corresponding authors.
[1]Code and models will be available at `https://github.com/microsoft/ToRA`.


Problem: Suppose that the sum of the squares of two complex numbers x and y is 7 and the sum of their cubes is 10. List all possible values for $x+y$, separated by commas.

We are given that $x^2+y^2=7$ and $x^3+y^3=10$. We can factor x^3+y^3 to get $(x+y)(x^2-xy+y^2)$.

Thus, we can write $10=(x+y)(7)$. It follows that $x+y$ must divide 10.

The only possibilities are that $x+y$ is $1, 2, 5$ or 10. ✗

```
import math

def solution():
    x = symbols('x'), y = symbols('y')
    x**2 + y**2 = 7
    x**3 + y**3 = 10

    x = (-1 + math.sqrt(17)) / 2
    y = (-1 - math.sqrt(17)) / 2

    # Get all possible values for x+y
    result = [x + y, -x - y]
```

```
>>> SyntaxError: cannot assign to expression.
```
✗

Let's use `Sympy` to calculate and print all possible values for $x+y$,

```
from sympy import symbols, Eq, solve, simplify

def possible_values():
    x, y = symbols('x y')
    eq1 = Eq(x**2 + y**2, 7)
    eq2 = Eq(x**3 + y**3, 10)
    solutions = solve((eq1,eq2), (x, y))
    sums = [simplify(sol[0] + sol[1]) for sol in solutions]
    return sums
print(possible_values())
```

```
>>> [-5, -5, 1, 1, 4, 4]
```

Removing duplicates, the possible values for $x+y$ are \boxed{-5, 1, 4}. ✓

Figure 2: Examples of three reasoning formats for mathematical reasoning: (a) Rationale-based methods (e.g., CoT prompting) generate step-by-step natural language rationales, (b) Program-based methods (e.g., PAL prompting) solve tasks with program synthesis, and (c) our proposed Tool-integrated Reasoning format interleaves rationales with program-based tool use. For brevity, we present a simple example of *single-round tool interaction*, where the model creates rationale r_1 for analysis, writes program a_1 to call an external solver, obtains the execution output o_1, and then generates rationale r_2 to finalize the answer.

1 Introduction

Large language models (LLMs), such as GPT-4 (OpenAI, 2023) and PaLM-2 (Anil et al., 2023), have demonstrated remarkable progress in a wide range of language tasks, particularly in the longstanding challenge of mathematical reasoning (Feigenbaum et al., 1963; Hosseini et al., 2014). However, open-source models, such as LLaMA-2 (Touvron et al., 2023a;b) and Falcon (Penedo et al., 2023), still struggle with advanced mathematical reasoning tasks.

Existing works improve mathematical performance of language models either with step-by-step natural language reasoning (Wei et al., 2022) as illustrated in Fig 2 (a), or by synthesizing and executing programs to obtain the answers (Gao et al., 2022; Chen et al., 2022), as depicted in Fig 2 (b). Both approaches exhibit complementary advantages. Natural language is suitable for semantic analysis, planning, and abstract reasoning (e.g., commonsense reasoning), but struggles with precise computation, symbolic manipulation, and algorithmic processing. Conversely, programs excel in rigorous operations, and can outsource intricate calculations to specialized tools like equation solvers.

To leverage the benefits of both natural language reasoning and program-based tool use, we train open-source models such as LLaMA-2 to reason in a way where natural language reasoning is interleaved with program-based tool use synergistically (as depicted in Fig 2 (c)), thereby largely reducing the gap with closed-source models like GPT-4 in mathematical reasoning. Specifically, we first design the interleaving format of reasoning, curate corresponding interactive tool-use trajectories for mathematical problems from the popular GSM8k (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) dataset, and then apply imitation learning on the high-quality annotations, leading to a better performance than any existing open-source model. Furthermore, since the curated data is far from exhausting all valid trajectories for a problem, relying solely on imitation learning restricts a model's output space, hindering the flexibility in exploring plausible trajectories during testing. To improve the diversity of plausible reasoning steps and mitigate improper tool-use behavior, we apply *output space shaping* which additionally trains the models on both self-sampled valid trajectories and invalid ones that have been corrected by a teacher model (e.g., a 34B model can serve as the teacher for a 7B model). *Output space shaping* significantly boosts reasoning performance, allowing

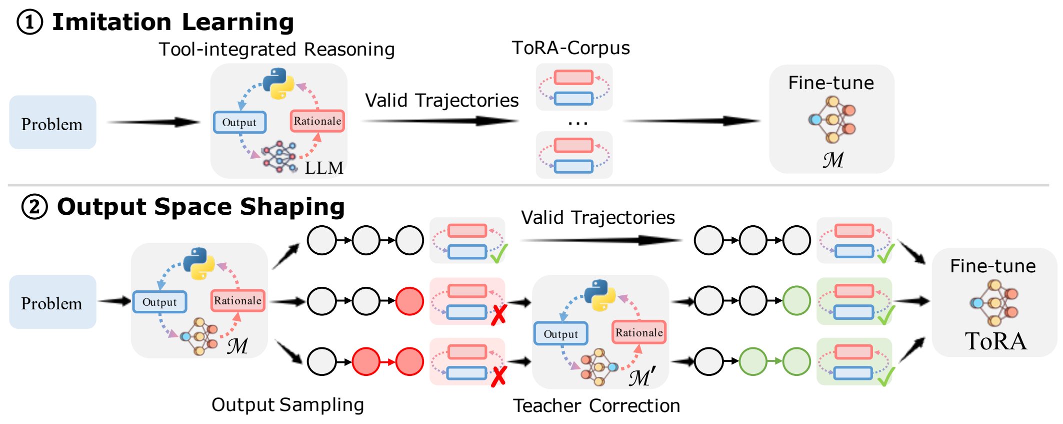


Figure 3: Training ToRA contains two steps. ① **Imitation Learning**: Prompt LLMs like GPT-4 to generate Tool-integrated Reasoning trajectories (ToRA-Corpus) and use this corpus to fine-tune a model $\mathcal{M}$; ② **Output Space Shaping**: Sample diverse tool-use trajectories with $\mathcal{M}$, keep the valid ones, correct the invalid ones with a teacher model $\mathcal{M}'$, and retrain $\mathcal{M}$ on the union of sampled valid trajectories, corrected ones, and the initial ToRA-Corpus to obtain ToRA.

open-source models to attain an accuracy exceeding 50% on the competition-level MATH dataset for the first time.

We evaluate the resulting suite of Tool-integrated Reasoning Agents (ToRA) ranging from 7B to 70B on 10 diverse mathematical reasoning datasets. As shown in Fig 1, ToRA series significantly outperform open-source models across all scales. Notably, on the competition-level MATH dataset, ToRA-7B outperforms the previous SoTA WizardMath-70B (Luo et al., 2023) by 22% absolute. ToRA-Code-34B beats GPT-4's CoT result (Bubeck et al., 2023) by 8.3% absolute (50.8% vs. 42.5%), and is competitive with GPT-4 solving problems with code (GPT-4-Code, 51.8%). In addition, we analyze the benefits and remaining challenges of tool interaction for mathematical reasoning, providing valuable insights for future work.

2 ToRA: Tool-Integrated Agents for Mathematical Reasoning

2.1 Overview

ToRA series solve challenging mathematical problems by leveraging both natural language reasoning and program-based tool use. As shown in Fig 2 (c), given a mathematical problem q, ToRA reasons with natural language, producing r_1. When reaching a point where program-based tool use is more appropriate for the subsequent task, e.g., equation solving, ToRA generates a program a_1 for tool use following natural language guidance r_1. The execution output o_1 will be fed to ToRA for subsequent processing including tool use adjustments, sub-tasks solving, or answer finalization. We repeat the process until the model places its answer within "`\boxed{}`". The resulting trajectory is denoted as $\tau = r_1 a_1 o_1 ... r_{n-1} a_{n-1} o_{n-1} r_n$, where r_n contains the answer.

Fig 3 presents the training pipeline of ToRA. We first collect interactive tool-use trajectories on popular mathematical datasets. We then apply imitation learning on the resulting annotations, as well as output space shaping to further refine models' reasoning behavior.

2.2 Collecting Interactive Tool-Use Trajectories

Existing mathematical reasoning datasets primarily contain annotations in either natural language or code, posing a challenge for training tool-integrated agents due to the absence of interactive tool-use annotations. To address this, we utilize GPT-4 to synthesize high-quality trajectories on the GSM8k and MATH training sets. We select GSM8k and MATH as they exhibit diverse reasoning patterns, spanning multiple domains and difficulty levels.

Algorithm 1 Inference of Tool-Integrated Reasoning

```
Require: problem $q$, model $\mathcal{G}$, prompt $\wp$, external tools $\mathcal{E}$, stop condition Stop(·), maximum iteration rounds $n$
 1: $\tau_0 \leftarrow$ ""                                              ▷ Trajectory Initialization
 2: for $i \leftarrow 1$ to $n$ do
 3:     $r_i \sim \mathbb{P}_{\mathcal{G}}(\cdot|\wp \oplus q \oplus \tau_{i-1})$                  ▷ Rationale Generation (Eq. 1)
 4:     if Stop($r_i$) then                                      ▷ Stopping Criteria
 5:         return $\tau_{i-1} \oplus r_i$
 6:     end if
 7:     $a_i \sim \mathbb{P}_{\mathcal{G}}(\cdot|\wp \oplus q \oplus \tau_{i-1} \oplus r_i)$             ▷ Program Generation (Eq. 2)
 8:     $o_i \leftarrow \mathcal{E}(a_i)$                                          ▷ Tool Execution
 9:     $\tau_i \leftarrow \tau_{i-1} \oplus r_i \oplus a_i \oplus o_i$                     ▷ Trajectory Update (Eq. 3)
10: end for
11: return $\tau_n$
```

Prompt Curation We compose instructions along with diverse few-shot examples, utilizing an interleaved format as depicted in Fig 2 (c). These examples showcase interactive tool usage trajectories, incorporating descriptive variable names and combined program outputs. Please refer to Appendix C for the assembled prompts.

Inference Procedure We follow Algorithm 1 and feed GPT-4 ($\mathcal{G}$) with the composed prompt $\wp$ to generate a tool-use trajectory τ for each question q from the training set. The trajectory is initialized as an empty string τ_0, for each interaction round i, we first generate a rationale:

$$r_i \sim \mathbb{P}_{\mathcal{G}}(\cdot|\wp \oplus q \oplus \tau_{i-1}) \tag{1}$$

where $\oplus$ means concatenation. If r_i includes an answer within "`\boxed{}`" (i.e., the stopping condition *Stop*(r_i)), we cease generation, otherwise the model continues to write a program for tool use:

$$a_i \sim \mathbb{P}_{\mathcal{G}}(\cdot|\wp \oplus q \oplus \tau_{i-1} \oplus r_i) \tag{2}$$

In line with Gou et al. (2023), if the model triggers the code execution stop words like "` ```output`", we supply it with the corresponding execution message and output o_i by calling tools with $o_i \leftarrow \mathcal{E}(a_i)$, facilitating the generation of subsequent steps. Then, we update the trajectory by concatenating it with the newly generated rationale r_i, program a_i, and output o_i:

$$\tau_i \leftarrow \tau_{i-1} \oplus r_i \oplus a_i \oplus o_i \tag{3}$$

We repeat the above interaction process until we reach the maximum rounds n.

Trajectory Sampling We set $n = 3$ and perform inference using GPT-4 with greedy decoding, retaining trajectories that yield correct answers. For questions where GPT-4 fails with greedy decoding, we apply nucleus sampling with a sample size of 10 and keep up to 4 valid trajectories per question. Ultimately, we successfully annotate trajectories for 98.2% of GSM8k questions and 83.1% of MATH questions. After filtering out invalid trajectories with tool-use errors or wrong answers, we obtain 16k annotations which constitute our dataset TORA-CORPUS. Table 1 compares TORA-CORPUS with recently proposed mathematical reasoning datasets, while Table 5 in the Appendix displays MATH annotation accuracy details.

2.3 TRAINING

Imitation Learning We apply imitation learning on TORA-CORPUS by minimizing negative log-likelihood loss on the trajectory τ conditioned on the problem q:

$$\mathcal{M} = \arg\min_{\theta} \sum_{q,\tau} \sum_{i=1}^{n-1} -\log \mathbb{P}_{\theta}(r_{i+1}a_{i+1}|q, r_1...o_i) \tag{4}$$

where $\mathcal{M}$ is the resulting model. After imitation learning, we can simply apply the same procedure in Algorithm 1 by setting prompt to empty $\wp =$ "" for inference. Imitation learning leads to state-of-the-art mathematical reasoning performance despite the small scale of TORA-CORPUS.

Table 1: Compared with mathematical reasoning datasets, ToRA-Corpus uniquely combines natural language rationales with program-based tool usage. Note that ToRA-Corpus only employ questions from the original training set of MATH and GSM8k.

Methods	**#Annotation**	**Tool**	**Interleaving**	**LLM Used**	**Source**
RFT (Yuan et al., 2023)	>100k	✗	✗	LLaMA-2	GSM8k
Open-Platypus Lee et al. (2023)	25k	✗	✗	GPT-4	11 datasets with MATH
WizardMath (Luo et al., 2023)	>96k	✗	✗	ChatGPT	MATH & GSM8k
Lila (Mishra et al., 2022)	134k	✓(PoT)	✗	-	20 datasets with MATH & GSM8k
MathInstruct (Yue et al., 2023)	260k	✓(PoT)	✗	GPT-4	14 datasets with MATH & GSM8k
ToRA-Corpus (ours)	16k	✓	✓	GPT-4	MATH & GSM8k

Output Space Shaping For each question, ToRA-Corpus mostly demonstrates only one valid interactive tool-use trajectory, which may restrict a model's output space, rendering it inflexible in exploring plausible trajectories during testing. We therefore propose *output space shaping* in order to encourage the diversity of plausible reasoning steps and reduce improper tool-use behavior.

To explore diverse valid trajectories, we apply nucleus sampling to imitation learning models $\mathcal{M}$ to sample 64 trajectories per training question q, following the inference procedure in Section 2.2. We retain valid trajectories with correct answers and no tool-use errors. As many samples are duplicates, to further improve diversity and in an attempt to correct models' improper behavior, we seek to leverage invalid trajectories as well. We observe that trajectories with wrong answers are mostly incorrect halfway (Li et al., 2023), and the preceding reasoning is still plausible; in other words, we can obtain valid trajectories by correcting the subsequent portions. Specifically, a wrong trajectory $\widetilde{\tau}$, when written in text, can be represented as a sequence of lines separated by line breaks, i.e., $\widetilde{\tau} = l_1 ... l_m$, where m is the total number of lines in $\widetilde{\tau}$. We enumerate possible preceding portions of wrong trajectories, i.e., $\widetilde{\tau}[:j] = l_1 ... l_j$, and leverage a teacher model $\mathcal{M}'$ to complete the subsequent steps with greedy decoding: $\tau \leftarrow \mathbb{P}_{\mathcal{M}'}(\cdot|q \oplus \widetilde{\tau}[:j])$ where we abuse the notation $\mathbb{P}_{\mathcal{M}'}(\cdot)$ to denote the interactive tool use process following Section 2.2. Finally, corrected trajectories as well as valid trajectory samples will be used for model training, thereby shaping the output space.

In our experiments, we always use CodeLLaMA-34B trained on ToRA-Corpus as the teacher model, and apply sampling with the CodeLLaMA series (ranging from 7B to 34B, with imitation learning). We obtain a total of 233k distinct valid trajectory samples and 69k corrected ones. From this combined dataset, we randomly select up to 4 trajectories per GSM8k and MATH problem, merge them with ToRA-Corpus, and then train all ToRA models on the resulting 69k annotations.

3 Experiments

3.1 Implementation Details

We fine-tuned LLaMA-2 (Touvron et al., 2023b) and CodeLLaMA (Rozière et al., 2023) series (ranging from 7B to 70B) using ToRA-Corpus with output space shaping, yielding the ToRA and ToRA-Code series respectively. We used a learning rate of 2e-5 by default except that we used 1e-5 for the 34B and 70B models. We set the global batch size to 128 and used a linear scheduler with a 3% warm-up period for 3 epochs. We trained all models with *DeepSpeed ZeRO Stage3* (Rajbhandari et al., 2021) and *Flash-Attention 2* (Dao, 2023). We used greedy decoding for all results, with the maximum sequence length set to 2,048 and the maximum number of tool executions set to 3.

3.2 Evaluation Setup

Datasets We evaluated models on GSM8k (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), along with 8 out-of-distribution datasets, namely GSM-Hard (Gao et al., 2022), SVAMP (Patel et al., 2021), ASDIV (Miao et al., 2020), TabMWP (Lu et al., 2023), SingleEQ, SingleOP, AddSub, and MultiArith (Koncel-Kedziorski et al., 2016), as illustrated in Table 4 in Appendix. The 10 assorted datasets collectively encompass mathematical problems spanning basic arithmetic to competition level, covering middle and high school curricula and various mathematical domains. The problem formats comprise tabular-based, free-form, and multiple-choice questions, ensuring a thorough assessment of the model's mathematical reasoning aptitude.

Table 2: Results on 10 mathematical reasoning tasks. MAWPS results are averaged over four tasks: Singleeq, Singleop, Addsub, and MultArith. Vanilla models are tested with CoT. The best results in each section are in blue, the second-best results are underlined, while the results of our best model are bolded. * ZS: Zero-shot inference without demonstrations.

Model	Size	Tools	ZS*	GSM8k	MATH	GSM-Hard	SVAMP	TabMWP	ASDiv	MAWPS	AVG
Used for training?				✓	✓	✗	✗	✗	✗	✗	
					Proprietary Models						
GPT-4	-	✗	✗	92.0	42.5	64.7	93.1	67.1	91.3	97.6	78.3
GPT-4 (PAL)	-	✓	✗	94.2	51.8	77.6	94.8	95.9	92.6	97.7	86.4
ChatGPT	-	✗	✗	80.8	35.5	55.9	83.0	69.1	87.3	94.6	72.3
ChatGPT (PAL)	-	✓	✗	78.6	38.7	67.6	77.8	79.9	81.0	89.4	73.3
Claude-2	-	✗	✗	85.2	32.5	-	-	-	-	-	-
PaLM-2	540B	✗	✗	80.7	34.3	-	-	-	-	-	-
					Open-Source Models						
LLaMA-2	7B	✗	✗	13.3	4.1	7.8	38.0	31.1	50.7	60.9	29.4
LLaMA-2 SFT	7B	✗	✓	41.3	7.2	16.1	31.9	27.8	47.4	60.0	33.1
LLaMA-2 RFT	7B	✗	✓	51.2	-	-	-	-	-	-	-
Platypus-2	7B	✗	✗	14.4	5.4	8.6	36.7	26.5	47.9	58.4	28.3
WizardMath	7B	✗	✓	54.9	10.7	20.6	57.3	38.1	59.1	73.7	44.9
CodeLLaMA (PAL)	7B	✓	✗	34.0	16.6	33.6	59.0	47.3	61.4	79.6	47.4
Toolformer†	7B	✓	✓	-	-	-	29.4	-	40.4	44.0	-
ToRA	7B	✓	✓	68.8	40.1	54.6	68.2	42.4	73.9	88.8	62.4
ToRA-Code	7B	✓	✓	**72.6**	**44.6**	**56.0**	**70.4**	**51.6**	**78.7**	**91.3**	**66.5 (+19)**
LLaMA-2	13B	✗	✗	24.3	6.3	13.6	43.1	39.5	56.3	70.4	36.2
LLaMA-2 SFT	13B	✗	✓	51.1	9.2	22.3	46.3	35.8	58.6	75.0	42.6
LLaMA-2 RFT	13B	✗	✓	55.3	-	-	-	-	-	-	-
Platypus-2	13B	✗	✗	23.7	7.1	14.3	50.7	45.3	55.1	69.6	38.0
WizardMath	13B	✗	✓	63.9	14.0	28.4	64.3	46.7	65.8	79.7	51.8
CodeLLaMA (PAL)	13B	✓	✗	39.9	19.9	39.0	62.4	59.5	65.3	86.0	53.1
ToRA	13B	✓	✓	72.7	43.0	57.3	72.9	47.2	77.2	91.3	65.9
ToRA-Code	13B	✓	✓	**75.8**	**48.1**	**60.5**	**75.7**	**65.4**	**81.4**	**92.5**	**71.3 (+18)**
LLaMA-1 RFT	34B	✗	✓	57.9	-	-	-	-	-	-	-
CodeLLaMA (PAL)	34B	✓	✗	53.3	23.9	49.4	71.0	63.1	72.4	91.5	60.7
ToRA-Code	34B	✓	✓	**80.7**	**50.8**	**63.7**	**80.5**	**70.5**	**84.2**	**93.3**	**74.8 (+14)**
LLaMA-2	70B	✗	✗	57.8	14.4	36.0	73.6	57.5	76.0	92.4	58.2
LLaMA-2 SFT	70B	✗	✓	69.3	14.9	39.0	64.0	53.0	71.3	84.8	56.6
LLaMA-2 RFT	70B	✗	✓	64.8	-	-	-	-	-	-	-
Platypus-2	70B	✗	✗	45.9	15.0	24.6	74.3	47.3	72.7	91.1	53.0
WizardMath	70B	✗	✓	81.6	22.7	50.3	80.0	49.8	76.2	86.2	63.8
LLaMA-2 (PAL)	70B	✓	✗	55.2	18.3	50.0	74.6	59.5	71.9	92.8	60.3
ToRA	70B	✓	✓	**84.3**	**49.7**	**67.2**	**82.7**	**74.0**	**86.8**	**93.8**	**76.9 (+13)**

Metrics We report accuracies of predicted answers. For numerical values, we perform rounding, while for expressions, we employ `sympy` [2] for parsing. Since the SingleEQ, SingleOP, AddSub, and MultiArith datasets focus on different aspects of basic arithmetic, we report their average results under the collective term MAWPS (Koncel-Kedziorski et al., 2016) for all methods.

3.3 Baselines

Proprietary Models We present results from an array of SoTA LLMs, such as OpenAI's GPT-4, ChatGPT (`gpt-3.5-turbo`), Google's PaLM-2, and Anthropic's Claude-2. By default, we report CoT prompting results, and include PAL (Gao et al., 2022) prompting results for selected models.

Open-Source Models *Base models* comprise LLaMA-2 and CodeLLaMA with CoT and PAL prompting. *Supervised Fine-Tuning (SFT)* employs CoT rationales from the original GSM8k and MATH dataset (15k samples) for fine-tuning. *Rejection sampling Fine-Tuning (RFT)* leverages multiple models to generate diverse reasoning paths for fine-tuning (Yuan et al., 2023). *WizardMath* augments data using ChatGPT, and conducts SFT and RLHF. *Platypus-2*, the top model on the LLM Leaderboard [3], is fine-tuned with Open-Platypus reasoning datasets (Lee et al., 2023). We also compare ToRA with Toolformer (Schick et al., 2023) which is a model trained to utilize calculators.

[2]`https://www.sympy.org`

[3]`https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`

Table 3: Results on MATH subtopics.

Model	Size	Tool	Intermediate Algebra	Precalculus	Geometry	Number Theory	Counting & Probability	Prealgebra	Algebra	Overall
						Proprietary Models				
ChatGPT (PAL)	-	✓	18.5	19.2	23.2	48.5	43.0	62.7	45.4	38.7
GPT-4 (PAL)	-	✓	32.8	29.3	38.0	58.7	61.0	73.9	59.1	51.8
						Open-Source Models				
WizarMath	7B	✗	6.2	6.0	6.5	7.6	9.5	18.1	16.3	11.2
ToRA-Code	7B	✓	**35.1 (+28.9)**	**31.0 (+25.0)**	**24.0 (+17.5)**	**50.7 (+43.1)**	**30.6 (+21.1)**	**55.0 (+36.9)**	**61.7 (+45.4)**	**44.6 (+33.4)**
w/o Shaping	7B	✓	29.7 (-5.4)	25.1 (-5.9)	17.7 (-6.3)	46.9 (-3.8)	32.3 (+1.7)	51.9 (-3.1)	55.7 (-6.0)	40.2 (-4.4)
w/o Rationale	7B	✓	25.5 (-9.6)	14.7 (-16.3)	15.4 (-8.6)	45.9 (-4.8)	29.7 (-0.9)	51.0 (-4.0)	52.4 (-9.3)	36.8 (-7.8)
WizarMath	13B	✗	6.4	6.6	11.5	9.6	11.0	28.5	21.1	15.0
ToRA-Code	13B	✓	**35.7 (+29.3)**	**31.1 (+24.5)**	**25.7 (+14.2)**	**55.6 (+46.0)**	**39.5 (+28.5)**	**58.7 (+30.2)**	**66.7 (+45.6)**	**48.1 (+33.1)**
w/o Shaping	13B	✓	32.8 (-2.9)	26.0 (-5.1)	24.0 (-1.7)	52.6 (-3.0)	38.4 (-1.1)	55.6 (-3.1)	61.2 (-5.5)	44.6 (-3.5)
w/o Rationale	13B	✓	27.1 (-8.6)	15.8 (-15.3)	16.3 (-9.4)	50.4 (-5.2)	36.9 (-2.6)	55.3 (-3.4)	56.5 (-10.2)	40.2 (-7.9)
ToRA-Code	34B	✓	**38.9**	**34.6**	**27.3**	**57.8**	**41.4**	**63.7**	**67.7**	**50.8**
w/o Shaping	34B	✓	34.0 (-4.9)	29.9 (-4.7)	24.6 (-2.7)	55.6 (-2.2)	41.6 (+0.2)	63.8 (+0.1)	61.4 (-6.3)	47.4 (-3.4)
w/o Rationale	34B	✓	28.3 (-10.6)	15.8 (-18.8)	18.0 (-9.3)	52.4 (-5.4)	40.7 (-0.7)	58.6 (-5.1)	57.5 (-10.2)	41.9 (-8.9)
WizarMath	70B	✗	9.1	13.4	16.9	16.5	19.2	42.7	35.0	24.1
ToRA	70B	✓	**37.1 (+28)**	**30.4 (+17)**	**30.1 (+13.2)**	**54.6 (+38.1)**	**40.3 (+21.1)**	**64.9 (+22.2)**	**66.6 (+31.6)**	**49.7 (+25.6)**
w/o Shaping	70B	✓	33.8(-3.3)	28.9(-1.5)	27.1(-3)	53.0(-1.6)	38.0(-2.3)	62.2(-2.7)	64.2(-2.4)	47.3(-2.4)
w/o Rationale	70B	✓	26.7(-10.4)	14.7(-15.7)	20.3(-9.8)	48.9(-5.7)	39.2(-1.1)	59.8(-5.1)	57.6(-9)	41.5(-8.2)

3.4 Main Results

Table 2 presents the results of ToRA on 10 mathematical datasets, highlighting the following salient observations: **(1)** Using interleaved formatting and output space shaping, ToRA consistently surpasses prior state-of-the-art open-source models across all scales, achieving 13% to 19% absolute improvements across 10 tasks. **(2)** ToRA-70B substantially outperforms ChatGPT with both CoT and PAL prompting on GSM8k (84.3% vs. 80.4%) and MATH (49.7% vs. 38.7%), while ToRA-Code-34B is competitive with GPT-4 solving competition-level MATH dataset with code (50.8% vs. 51.8%). **(3)** The accuracy of ToRA-Code is about 5% higher than ToRA of the same size, demonstrating that continued training on code data significantly benefits program-based tool use. **(4)** While rationale-based fine-tuning negatively affects out-of-distribution generalization, ToRA displays superior generalization. For instance, WizardMath-70B underperforms the base model on TabMWP (49.8% vs. 57.5%), while ToRA-70B effectively generalizes to this tabular reasoning task (74.0%). **(5)** ToRA attains fast zero-shot inference speed, averaging 1.02 tool interaction rounds per problem, while effectively addressing problems that require interactive tool utilization.

3.5 Ablation Study

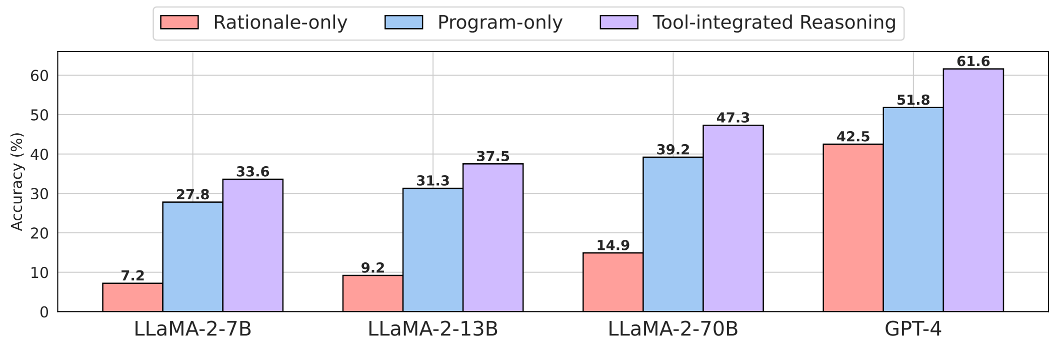


Figure 4: Comparison of three formats: (1) Rationale-only: step-by-step natural language reasoning like CoT; (2) Program-only: solving problems with programs like PAL; (3) Tool-integrated Reasoning used by ToRA: interweaving rationale and program execution to solve problems. We evaluated GPT-4 with few-shot prompting. We trained LLaMA-2 models to reason in the three types of formats, respectively. For a fair comparison, we *do not apply output space shaping* for all LLaMA-2 models.

3.5.1 Comparisons of Formatting

To evaluate the efficacy of the reasoning format adopted by ToRA which interleaves rationales with programs, we compared it with Rationale-only and Program-only formats using GPT-4 and LLaMA-2 trained with the same size of data from MATH. As shown in Fig 4, the ToRA method consistently surpasses Rationale-only and Program-only approaches. Remarkably, Using LLaMA-2, the ToRA method achieves substantial improvements of 29.0% and 6.7% over Rationale-only and Program-only, respectively. With the closed-source GPT-4, the improvements are 19.1% and 9.8%, respectively. This emphasizes the effectiveness of integrating natural language rationales with programs.

3.5.2 Effects of Output Space Shaping

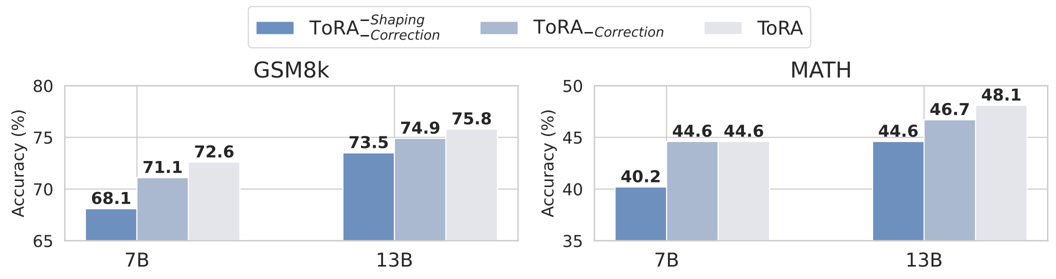


Figure 5: Ablation on output space shaping strategies using CodeLLaMA: (1) ToRA$^{-\text{Shaping}}_{-\text{Correction}}$ is trained on ToRA-Corpus without shaping. (2) ToRA$_{-\text{Correction}}$ employs only the sampling strategy for shaping, trained with up to 4 additional valid trajectory samples per problem. (3) ToRA utilizes both the sampling and correction, also trained with up to 4 additional trajectories per problem.

We assess the effectiveness of the output space shaping strategies presented in Section 2.3, specifically sampling and correction. As shown in Fig 5: (1) Output space shaping yields a considerable average improvement of 3.4% and 4.0% absolute for GSM8k and MATH, respectively, with greater benefits for smaller models; (2) Applying the sampling strategy results in a 2.7% absolute improvement on average, while additionally incorporating correction offers a modest yet significant average improvement of 0.8% to 1.2% absolute; (3) Output space shaping benefits even the largest model ToRA-70B, with a notable improvement from 47.3% to 49.7% on MATH. These findings highlight the effectiveness of our shaping strategies across different model sizes and datasets.

3.6 Analysis

We investigate the benefits, detailed patterns, and remaining challenges of tool interaction for mathematical reasoning on the challenging MATH dataset. Performance breakdowns on all subtopics of MATH are reported in Table 3.

Benefits from Tool-Integration for MATH Sub-topics As shown in Table 3, ToRA outperforms WizardMath by around 45% in Algebra and Number Theory, which is attributed to stimulating and shaping tool-use behavior. Problems from the two sub-topics typically need intricate computation and data manipulation. Algebra mainly focuses on solving equations and application problems, while many Number Theory problems can be tackled using brute-force approaches through code.

Patterns of Library Usage for Problem Solving Fig 6 presents the most frequently used libraries for different sub-topics and the corresponding accuracies of their solutions. Tool-use behavior on different mathematical areas demonstrates distinct patterns. `sympy` and its internal `solvers` are primarily employed for algebra-related topics. Precalculus exhibits extensive matrix operations via `matrices`, resulting in a high accuracy. Number Theory depends on `algorithms` like `gcd` and `lcm`. Geometry mainly uses the `rational` library for fraction-based computations, while the application of other tools is limited, signifying the potential for improvement.

Detailed Impact of Rationale on Different Topics Table 3 shows that using an interleaved format, in contrast to merely writing the program, leads to significant improvements across all subtopics, especially in Precalculus, Algebra, and Geometry, where notable increases range from 8.6% to 18.8%.

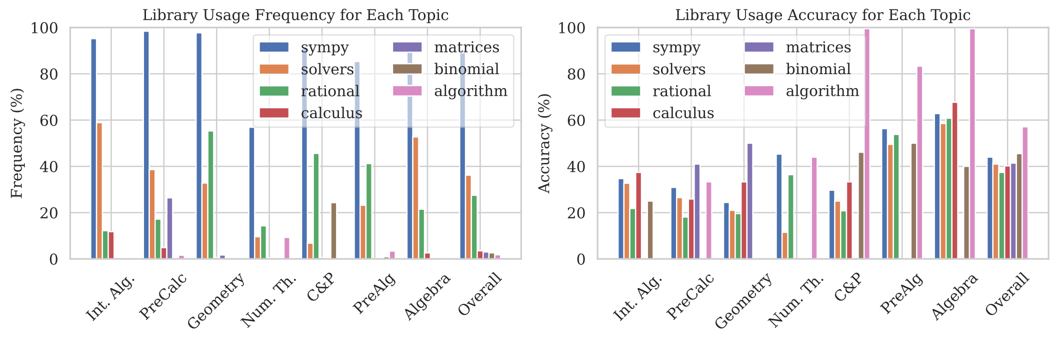


Figure 6: Library usage frequency and accuracy on each sub-topic of MATH.

Appendix D.1 provides representative examples demonstrating how the rationale aids in planning, multi-round self-correction, and finalizing answers.

Remaining Challenges in Mathematical Reasoning for ToRA Although ToRA has made notable progress in various mathematical domains, substantial improvements are still needed in topics like Geometry, Precalculus, and Intermediate Algebra. In Geometry, as illustrated by failure cases in Listing 6 in Appendix, a deeper understanding of geometric space is essential, encompassing visual modalities and interactions with images for auxiliary information, while incorporating computational tools like `SymPy` offers limited benefits. For Intermediate Algebra and Precalculus problems, as shown in Listing 5, direct brute-force solutions are often infeasible, resulting in timeout exceptions. Addressing these challenges requires complex symbolic reasoning over algebraic expressions and the given conditions, along with sophisticated problem-solving and proof techniques involving forward and backward reasoning, as well as result verification.

4 Related Works

Mathematical Reasoning Recent research has greatly improved reasoning in LLMs with step-by-step natural language reasoning (Wei et al., 2022; Zhou et al., 2023; Zhu et al., 2023; Huang et al., 2022; Liang et al., 2023). However, natural language reasoning struggles with complex computations and symbolic manipulations. To overcome the limitations, recent research has exploited tools like calculators (Cobbe et al., 2021; Shao et al., 2022), code interpreters (Mishra et al., 2022), and symbolic solvers (Zhang et al., 2023). Program-based methods (Gao et al., 2022; Chen et al., 2022; Shao et al., 2023a) transform reasoning tasks into program synthesis tasks, thus offering complementary advantages over natural language reasoning, but they face challenges in nuanced reasoning, planning, and error handling (Gou et al., 2023), where natural language reasoning should be more suitable.

Tool-Augmented Language Models Augmenting LLMs with tools can largely alleviate LLMs' limitations and improve reasoning and generation performance (Parisi et al., 2022; Mialon et al., 2023; Yao et al., 2023). Recent work demonstrates the benefits of integrating retrievers (Borgeaud et al., 2022; Shao et al., 2023b), search engines (Nakano et al., 2021), and multi-tool approaches (Schick et al., 2023; Paranjape et al., 2023; Gou et al., 2023) to improve generation.

Knowledge Distillation Knowledge distillation (KD) transfers knowledge from teacher models to student models (Buciluǎ et al., 2006; Hinton et al., 2015). Using LLM-generated trajectories for fine-tuning is a form of KD (Fu et al., 2023; Taori et al., 2023; Peng et al., 2023; Ho et al., 2023). Our proposed ToRA shows that learning interactive tool-use trajectories is a promising direction to adapt language models to reasoning tasks.

5 Conclusion

This paper presents ToRA, a series of novel Tool-integrated Reasoning Agents that synergistically combines natural language rationale with program-based tool-use for mathematical problem solving. Our approach demonstrates the potential of integrating external tools in the reasoning process, enabling language models to effectively tackle complex quantitative tasks. ToRA achieves state-of-the-art performance on 10 diverse mathematical reasoning tasks, substantially outperforming existing rationale-based and program-based approaches. Furthermore, our systematic analysis of the benefits and remaining challenges of tool interaction provides valuable insights for future research, paving the way for the development of more advanced and versatile reasoning agents.

6 Author Contributions

Zhibin Gou proposed the interleaved tool-use format of ToRA and curated ToRA-Corpus dataset, implemented the training and evaluation pipeline, conducted experiments and analysis on all datasets, implemented baselines, and was a main contributor to the paper writing. Zhihong Shao proposed the project, conducted preliminary experiments, proposed and implemented the training and evaluation pipelines, proposed and trained all ToRA models with output space shaping as well as ToRA variants in the ablation study, designed and oversaw experimental analysis, and contributed to many parts of the paper writing. Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen provided research mentorship, oversaw project coordination, and advised and contributed to many parts of the writing.

References

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with GPT-4. *CoRR*, abs/2303.12712, 2023. doi: 10.48550/arXiv.2303.12712. URL `https://doi.org/10.48550/arXiv.2303.12712`.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

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

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. Training verifiers to solve math word problems, 2021. URL `https://arxiv.org/abs/2110.14168`.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. 2023.

Edward A Feigenbaum, Julian Feldman, et al. *Computers and thought*, volume 7. New York McGraw-Hill, 1963.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. Specializing smaller language models towards multi-step reasoning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10421–10430. PMLR, 2023. URL `https://proceedings.mlr.press/v202/fu23d.html`.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*, 2022.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*, 2023.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14852–14882, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.830. URL `https://aclanthology.org/2023.acl-long.830`.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 523–533, 2014.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. *CoRR*, abs/2210.11610, 2022. doi: 10.48550/arXiv.2210.11610. URL `https://doi.org/10.48550/arXiv.2210.11610`.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 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*, pp. 1152–1157, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1136. URL `https://aclanthology.org/N16-1136`.

Ariel N Lee, Cole J Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of llms. *arXiv preprint arXiv:2308.07317*, 2023.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 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)*, pp. 5315–5333, 2023.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*, 2023.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=DHyHRBwJUTN`.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*, 2023.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 975–984, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.92. URL `https://aclanthology.org/2020.acl-main.92`.

Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, and Ashwin Kalyan. Lila: A unified benchmark for mathematical reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

OpenAI. Gpt-4 technical report, 2023.

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

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

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 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*, pp. 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL `https://aclanthology.org/2021.naacl-main.168`.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023. URL `https://arxiv.org/abs/2306.01116`.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.

Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–14, 2021.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

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

Zhihong Shao, Fei Huang, and Minlie Huang. Chaining simultaneous thoughts for numerical reasoning. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 2533–2547. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-emnlp.187. URL `https://doi.org/10.18653/v1/2022.findings-emnlp.187`.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Synthetic prompting: Generating chain-of-thought demonstrations for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 30706–30775. PMLR, 2023a. URL `https://proceedings.mlr.press/v202/shao23a.html`.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. *CoRR*, abs/2305.15294, 2023b. doi: 10.48550/arXiv.2305.15294. URL `https://doi.org/10.48550/arXiv.2305.15294`.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023b. doi: 10.48550/arXiv.2307.09288. URL `https://doi.org/10.48550/arXiv.2307.09288`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=_VjQlMeSB_J`.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=WE_vluYUL-X`.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.

Beichen Zhang, Kun Zhou, Xilin Wei, Wayne Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. Evaluating and improving tool-augmented computation-intensive math reasoning. *arXiv preprint arXiv:2306.02408*, 2023.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/pdf?id=WZH7099tgfM`.

Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Yongfeng Huang, Ruyi Gan, Jiaxing Zhang, and Yujiu Yang. Solving math word problems via cooperative reasoning induced language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4471–4485, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.245. URL `https://aclanthology.org/2023.acl-long.245`.

A Evaluation Datasets

Table 4: Statistics and examples of the 10 evaluation datasets. In the main result table, we present the average accuracy of SingleEq, SingleOp, AddSub, and MultiArith under the collective name MAWPS.

Dataset	OOD?	#Samples	Example Problem
GSM8k (Cobbe et al., 2021)	IND	1319	The ice cream parlor was offering a deal, buy 2 scoops of ice cream, get 1 scoop free. Each scoop cost \$1.50. If Erin had \$6.00, how many scoops of ice cream should she buy?
MATH (Hendrycks et al., 2021)	IND	5000	For a constant c, in cylindrical coordinates (r, θ, z), find the shape described by the equation $$z = c.$$ (A) Line (B) Circle (C) Plane (D) Sphere (E) Cylinder (F) Cone. Enter the letter of the correct option.
GSM-Hard (Gao et al., 2022)	OOD	1319	Jean has 30 lollipops. Jean eats 8714250 of the lollipops. With the remaining lollipops, Jean wants to package 8714250 lollipops in one bag. How many bags can Jean fill?
SVAMP (Patel et al., 2021)	OOD	1000	During summer break 819058 kids from Lawrence county go to camp and the other 668278 kids stay home. How many more kids spent their summer break at the camp compared to those who stayed home?
ASDiv (Miao et al., 2020)	OOD	2215	Mrs. Hilt saw an iPod for sale. The price tag said the iPod cost \$128, but a sign announced that it was on sale for "35% off." How much would the iPod cost after the discount?
TabMWP (Lu et al., 2023)	OOD	1000	Stem / Leaf: 2 / 3, 6, 7, 8, 8; 3 / 0, 7, 9; 4 / 1, 5; 5 / ; 6 / 2, 3, 3, 4, 8, 8; 7 / 3, 4, 4, 7, 9; 8 / 5, 5. Read the table regarding "eight lifting results (lbs)". Mr. Morrison, a P.E. teacher, wrote down how much weight each of his students could lift. How many people lifted at least 28 pounds?
SingleEq (Koncel-Kedziorski et al., 2016)	OOD	508	Alyssa's dog had puppies. She gave 7 to her friends.She now has 5 puppies left. How many puppies did she have to start with?
SingleOp (Koncel-Kedziorski et al., 2016)	OOD	562	Rachel removes 47 bottle caps from a jar. There were originally 87 bottle caps in the jar. How many bottle caps are left in the jar?
AddSub (Koncel-Kedziorski et al., 2016)	OOD	395	Sam went to 14 football games this year. He went to 29 games last year. How many football games did Sam go to in all?
MultArith (Koncel-Kedziorski et al., 2016)	OOD	600	Paige had 43 math problems and 12 science problems for homework. If she finished 44 of the problems at school, how many problems did she have to do for homework?

We present statistics and examples of the ten evaluation datasets in Table 4.

B Additional Experiments

B.1 Accuracies of Closed-Source Models on MATH

Table 5: Accuracies of ChatGPT and GPT-4 on the MATH dataset, with breakdown w.r.t. different mathematical subjects. We apply PAL prompting and the Tool-integrated Reasoning method used by ToRA to the two closed-source models.

Model	**Tool**	**Intermediate Algebra**	**Precalculus**	**Geometry**	**Number Theory**	**Counting & Probability**	**Prealgebra**	**Algebra**	**Overall**
					Test Set				
ChatGPT (PAL)	✓	18.5	19.2	23.2	48.5	43.0	62.7	45.4	38.7
GPT-4 (PAL)	✓	32.8	29.3	38.0	58.7	61.0	73.9	59.1	51.8
GPT-4 (Tool-integrated Reasoning)	✓	40.0	37.2	44.1	68.9	67.3	82.2	75.8	61.6
					Training Set				
GPT-4 (Tool-integrated Reasoning)	✓	51.0	51.5	42.5	77.4	72.2	89.8	85.1	64.3
w/ best@10	✓	72.9	70.0	58.9	91.6	81.7	95.5	96.3	83.1

Table 5 presents the detailed accuracies of GPT-4 on the MATH dataset. The Tool-integrated Reasoning method used by ToRA significantly outperforms PAL prompting when directly applied to the closed-source GPT-4, further demonstrating the benefits of synergizing natural language reasoning and program-based tool use.

As described in Section 2.2, we annotated interactive tool-use trajectories for the training questions from MATH with GPT-4. GPT-4 achieves a success rate below 65% using greedy decoding. As MATH is originally annotated with natural language rationales, to improve the annotation success rate, we tried to provide GPT-4 with the rationales as hints. However, when using this method, GPT-4 tends to replicate the hints and ignore tool-use outputs especially when the outputs are inconsistent with the hints, thus failing to produce high-quality trajectories. Hence, we deferred the utilization of the already-annotated natural language rationales for future investigations. Instead, we employed nucleus sampling to recall valid trajectories for questions that remained unsolved through greedy decoding. This approach significantly boosted annotation accuracy to 83.1%.

B.2 Effects of # Valid Trajectories for Output Space Shaping

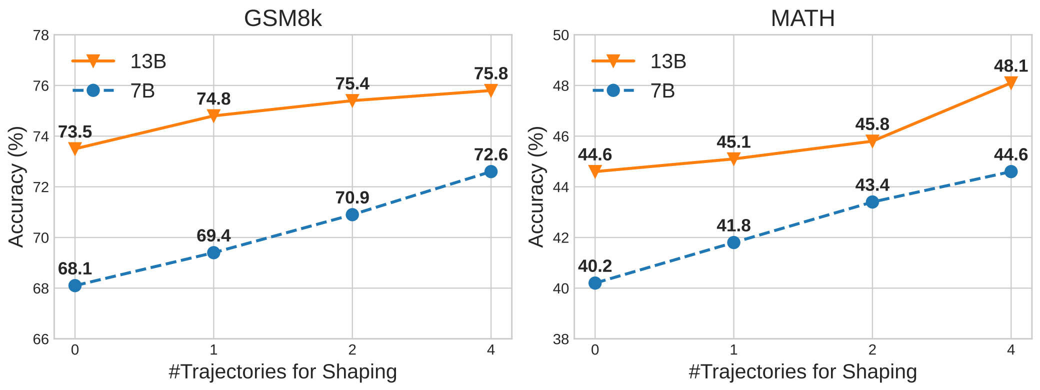


Figure 7: Effects of using difference numbers of additional valid trajectories per question for output space shaping.

As shown in Fig 7, it is beneficial to increase the number of additional valid trajectories for output space shaping.

C PROMPTS

We present instructions and example few-shot prompts of Tool-integrated Reasoning for querying GPT-4.

Listing 1: Prompts of Tool-integrated Reasoning for GPT-4.

~~~
Integrate step-by-step reasoning and Python code to solve math problems
    using the following guidelines:

- Analyze the question and write functions to solve the problem; the
    function should not take any arguments.
- Present the final result in LaTeX using a '\boxed{}' without any units.
- Utilize the 'pi' symbol and 'Rational'' from Sympy for $\pi$ and
    fractions, and simplify all fractions and square roots without
    converting them to decimal values.

Here are some examples you may refer to:

---

Question: Convert the point $(0, -3 \sqrt{3}, 3)$ in rectangular
    coordinates to spherical coordinates. Enter your answer in the form $
    (\rho,\theta,\phi),$ where $\rho > 0,$ $0 \le \theta < 2 \pi,$ and $0
     \le \phi \le \pi.$

Solution:
To convert the point $(x, y, z)$ in rectangular coordinates to spherical
    coordinates $(\rho, \theta, \phi)$, we use the following formulas:
1. $\rho = \sqrt{x^2 + y^2 + z^2}$
2. $\theta = \arctan{\frac{y}{x}}$
3. $\phi = \arccos{\frac{z}{\rho}}$

Let's write a function:

```python
from sympy import sqrt, atan2, acos, pi

def rectangular_to_spherical():
   x, y, z = 0, -3*sqrt(3), 3
   rho = sqrt(x**2 + y**2 + z**2)
   theta = atan2(y, x)
   phi = acos(z/rho)
   return rho, theta, phi

spherical_coordinates = rectangular_to_spherical()
print(spherical_coordinates)
```
```output
(6, -pi/2, pi/3)
```

The required form of the answer is $(\rho, \theta, \phi)$, where $\rho >
    0$, $0 \le \theta < 2\pi$, and $0 \le \phi \le \pi$. Since our
    initial output has $\theta = -\pi/2$, which doesn't meet the
    condition for $\theta$, we add $2\pi$ to adjust its value.

Hence, the spherical coordinates for the point $(0, -3\sqrt{3}, 3)$ are $
    \boxed{(6, \frac{3\pi}{2}, \frac{\pi}{3})}$.

---

Question: Compute $1011_2 + 101_2 - 1100_2 + 1101_2$. Express your answer
     in base 2.
~~~

~~~
Solution:
Let's write a single python function for binary computation:

```python
def binary_sum_diff():
   # Convert binary to decimal
   num1 = int("1011", 2)
   num2 = int("101", 2)
   num3 = int("1100", 2)
   num4 = int("1101", 2)

   # Perform the operations
   result = num1 + num2 - num3 + num4

   # Convert the result back to binary
   result_binary = format(result, "b")

   return result_binary

result = binary_sum_diff()
print(result)
```
```output
10001
```
The answer expressed in base 2 is $\boxed{10001_2}$.

---

Question: Find the matrix that corresponds to projecting onto the vector
    $\begin{pmatrix} 2 \\ -3 \end{pmatrix}.$

Solution:
```python
from sympy import Matrix, simplify

def projection_matrix():
   v = Matrix([2, -3])
   P = (v * v.T) / (v.T * v)[0]

   # Simplify the elements
   P = P.applyfunc(simplify)

   return P

result = projection_matrix()
print(result)
```
```output
Matrix([[4/13, -6/13], [-6/13, 9/13]])
```

The matrix is $\boxed{\begin{pmatrix} \frac{4}{13} & -\frac{6}{13} \ -\
    frac{6}{13} & \frac{9}{13} \end{pmatrix}}.$

---

Question: Solve the inequality
\[\frac{x^2 - 25}{x + 5} < 0.\]

Solution:
Let's use `sympy` to solve the inequality:

```python
from sympy import symbols, simplify
~~~

```
from sympy.solvers.inequalities import solve_univariate_inequality
from sympy.core.relational import LessThan

def solve_inequality():
   x = symbols('x')
   expression = (x**2 - 25) / (x + 5)
   inequality = LessThan(expression, 0)
   solution = solve_univariate_inequality(inequality, x, relational=False
       )
   simplified_solution = simplify(solution)

   return simplified_solution

result = solve_inequality()
print(result)
```
```output
Union(Interval.open(-oo, -5), Interval.Lopen(-5, 5))
```
The answer is $\boxed{(-\infty,-5)\cup(-5,5)}$

```

D EXAMPLES

D.1 SUCCESS CASES

Listing 2: Success case for TORA: **Rationale for planning**.

```
Question: What real values of $x$ are not in the domain of $f(x)=\frac
    {1}{|x^2-6x+8|+|x^2+x-6|}$?

Solution:
The domain of the function is all real numbers, so we need to find the
    values of x for which the denominator of the function is zero.
The denominator of the function is the sum of the absolute values of the
    two terms inside the absolute value.
Each of these terms is a quadratic in x, and the roots of a quadratic
    equation are real unless the equation is identically zero.
So, we need to find the values of x for which the two quadratic equations
     are identically zero.

```python
from sympy import symbols, Abs, solveset, S
from sympy.core.relational import Ne

def non_domain_values():
   x = symbols('x')
   # These are the values of x for which the function is not defined
   roots1 = solveset(Ne(x**2 - 6*x + 8, 0), x, domain=S.Reals)
   roots2 = solveset(Ne(x**2 + x - 6, 0), x, domain=S.Reals)
   # The roots of a quadratic equation are always in the domain of the
       function, so we need to find the values of x that are not in the
       union of the roots
   non_domain_values = S.Reals - (roots1.union(roots2))

   return non_domain_values

result = non_domain_values()
print(result)
```
```output
{2}
```

```
```
The real value of $x$ that is not in the domain of the function is $\
    boxed{2}$.
```

Listing 3: Success case for ToRA: **Self-Correcting Errors with tool feedback**.

```
Question: The difference between two perfect squares is 133. What is the
    smallest possible sum of the two perfect squares?

Solution:
We can solve this problem by noting that any perfect square can be
    written as $(a + b)^2$ or $(a - b)^2$ for some integers $a$ and $b$.
    So we can rewrite the question as the following problem: Find the
    minimum possible value for $(a + b)^2 + (a - b)^2$ given that $(a + b)
    ^2 - (a - b)^2 = 133$.

```python
from sympy import symbols, Eq, solve

def min_sum_perfect_squares():
   """
   Find the smallest possible sum of two perfect squares
   given that the difference between them is 133.

   Returns:
      Integer: The minimum possible sum of the two perfect squares.
   """
   a, b = symbols('a b')

   # Equation for the difference of the squares
   diff_eq = Eq((a + b)**2 - (a - b)**2, 133)

   # Solve the equation for b
   b_solutions = solve(diff_eq, b)

   # Calculate the sum of the squares for each solution of b
   sum_squares = [(a + b_sol)**2 + (a - b_sol)**2 for b_sol in
       b_solutions]

   # Find the minimum possible sum of the squares
   min_sum_squares = min(sum_squares)

   return min_sum_squares

result = min_sum_perfect_squares()
print(result)
```
```output
(a - 133/(4*a))**2 + (a + 133/(4*a))**2
```

The problem seems to be more complex than anticipated and the solution
    was incorrect.
Since the difference between two squares can be factored, we can write:

$x^2 - y^2 = (x + y)(x - y)$

$133$ has two pairs of factors: 1 and 133, and 7 and 19.

So, either $x+y = 133$ and $x-y = 1$, or $x+y = 19$ and $x-y = 7$.

- For (133, 1), $x = \frac{133 + 1}{2} = 67$ and $y = \frac{133 - 1}{2} =
     66$.
- For (19, 7), $x = \frac{19 + 7}{2} = 13$ and $y = \frac{19 - 7}{2} = 6$.
```
```

~~~
(13, 6) result in the smallest sum of the two perfect squares.

So, let's modify the previous solution:

```python
def min_sum_perfect_squares():

   min_squares = [13**2, 6**2]
   min_sum_squares = sum(min_squares)

   return min_sum_squares
~~~
```output
205
```

The smallest possible sum of the two perfect squares is $\boxed{205}$.
```

Listing 4: Success case for TORA: **Finalizing answer**.

```
Question:
Compute \[\sum_{k = 1}^\infty \frac{k + 2}{k! + (k + 1)! + (k + 2)!}.\]

Solution:
```python
from sympy import symbols, Sum, factorial, oo, simplify

def infinite_sum():
   expr = (k + 2) / (factorial(k) + factorial(k + 1) + factorial(k + 2))
   sum_expr = Sum(expr, (k, 1, oo))
   result = sum_expr.doit()

   return result

result = infinite_sum()
print(result)
```
```output
Sum((k + 2)/(factorial(k) + factorial(k + 1) + factorial(k + 2)), (k, 1,
    oo))
```
The sum does not converge to a simple expression, so the answer is $\
    boxed{\sum_{k = 1}^\infty \frac{k + 2}{k! + (k + 1)! + (k + 2)!}}$.
```

## D.2 FAILURE CASES

Listing 5: Failure case for TORA: **Failed fallback from program timeout to rationale-based solution**.

```
Question: For some positive integer $n,$ $0 < n < 180,$
\[\csc (2^3)^\circ + \csc (2^4)^\circ + \csc (2^5)^\circ + \dots + \csc
    (2^{2019})^\circ = \sec n^\circ.\]Find $n.$

Solution:
```python
from sympy import symbols, solve, csc, sec, pi

def find_n():
   n = symbols('n')
   lhs = sum(csc(2**i) for i in range(3, 2020))
   rhs = sec(n)
   equation = lhs - rhs
```

```
    solution = solve(equation, n)

    # Filter out solutions that are not in the interval (0, 180)
    solution = [sol.evalf() for sol in solution if 0 < sol.evalf() < 180]

    return solution[0]

result = find_n()
print(result)
```
```output
timeout_decorator.timeout_decorator.TimeoutError: 'Timed Out'
```
The solution to this problem is not straightforward to solve using sympy
    due to the complexity of the equation and the large number of terms
    in the sum.

However, we can solve this problem using a different approach. The sum on
     the left hand side is a telescoping sum, which means that many of
    the terms cancel out. After cancelling, we are left with $\csc
    (2^{2019}) - \csc(2^3)$.

Since $2^{2019}$ is very close to $2^{2020}$, we can approximate $\csc
    (2^{2019})$ as $\csc(2^{2020})$.

Then, the equation becomes $\csc(2^{2020}) - \csc(2^3) = \sec(n)$, which
    simplifies to $\csc(2^{2020}) = \sec(n) + \csc(2^3)$.

Since $\csc(2^3)$ is very close to 1, we can approximate $\csc(2^3)$ as 1.

Then, the equation becomes $\csc(2^{2020}) = \sec(n) + 1$, which
    simplifies to $\csc(2^{2020}) - 1 = \sec(n)$.

Since $\csc(2^{2020})$ is very close to 1, we can approximate $\csc
    (2^{2020})$ as 1.

Then, the equation becomes $1 - 1 = \sec(n)$, which simplifies to $\sec(n
) = 0$.

Since $0 < n < 180$, the only solution is $n = 90$. So, the value of $n$
    is $\boxed{90}$.

Ground Truth: 82
```

Listing 6: Failure case for ToRA: **Geometric problems**.

```
Question: A square has a side length of 10 inches. Congruent isosceles
    right triangles are cut off each corner so that the resulting octagon
     has equal side lengths. How many inches are in the length of one
    side of the octagon? Express your answer as a decimal to the nearest
    hundredth.

[asy]
size(150);
draw((0,0)--(10,0)--(10,10)--(0,10)--(0,0),linewidth(0.7));
draw((5*sqrt(2)/(1 + sqrt(2)),0)--(0,5*sqrt(2)/(1 + sqrt(2))),linewidth
    (0.7));
draw((10 - (5*sqrt(2)/(1 + sqrt(2))),0)--(10,5*sqrt(2)/(1 + sqrt(2))),
    linewidth(0.7));
draw((0,(10-(5*sqrt(2)/(1+sqrt(2)))))--(5*sqrt(2)/(1 + sqrt(2)),10),
    linewidth(0.7));
draw((10 - (5*sqrt(2)/(1 + sqrt(2))),10)--(10,10 - 5*sqrt(2)/(1 + sqrt(2)
    )),linewidth(0.7));
```
```

~~~
[/asy]

Solution:
The side length of the octagon is the hypotenuse of the right triangle
    formed by the side of the square and the diagonal of the octagon
The hypotenuse is the sum of the side length of the square and the length
     of the diagonal of the octagon
The length of the diagonal of the octagon is 5*sqrt(2)/(1 + sqrt(2))

```python
from sympy import sqrt, Rational

def octagon_side_length():
   square_side_length = 10
   octagon_side_length = square_side_length + 5*sqrt(2)/(1 + sqrt(2))

   return round(octagon_side_length, 2)

result = octagon_side_length()
print(result)
```
```output
12.93
```
The length of one side of the octagon is $\boxed{12.93}$ inches.

**Ground Truth**:
Recall that the ratio of the hypotenuse to the leg of an isosceles right
    triangle is $\sqrt{2}$. Looking at the removed triangles, we see that
     the three segments that make up the side of square are $s/\sqrt{2}$,
     $s$, and $s/\sqrt{2}$, where $s$ is the side length of the octagon.
    Setting the sum of these three side lengths equal to 10 inches, we
    find \begin{align*}
\frac{s}{\sqrt{2}}+s+\frac{s}{\sqrt{2}}&=10 \implies \\
s\left(\frac{1}{\sqrt{2}}+1+\frac{1}{\sqrt{2}}\right)&=10 \implies \\
s(\sqrt{2}+1)&=10 \implies \\
s&=\frac{10}{\sqrt{2}+1}=10(\sqrt{2}-1),
\end{align*} where we have rationalized the denominator twice: \[
\frac{1}{\sqrt{2}}=\frac{1}{\sqrt{2}}\cdot \frac{\sqrt{2}}{\sqrt{2}}=\
    frac{\sqrt{2}}{2},
\] and \[
\frac{10}{\sqrt{2}+1}=\frac{10}{\sqrt{2}+1}\cdot\frac{\sqrt{2}-1}{\sqrt
    {2}-1}=10(\sqrt{2}-1).
\] To the nearest hundredth, $s=\boxed{4.14}$ inches.

[asy]
size(150);
defaultpen(linewidth(0.7)+fontsize(10));
real s = 10/(1+sqrt(2));
draw((0,0)--(10,0)--(10,10)--(0,10)--(0,0));
draw((5*sqrt(2)/(1 + sqrt(2)),0)--(0,5*sqrt(2)/(1 + sqrt(2))));
draw((10 - (5*sqrt(2)/(1 + sqrt(2))),0)--(10,5*sqrt(2)/(1 + sqrt(2))));
draw((0,(10-(5*sqrt(2)/(1+sqrt(2)))))--(5*sqrt(2)/(1 + sqrt(2)),10));
draw((10 - (5*sqrt(2)/(1 + sqrt(2))),10)--(10,10 - 5*sqrt(2)/(1 + sqrt(2)
    )));

label("$s$",(10-s/(2*sqrt(2)),10-s/(2*sqrt(2))),SW);
label("$\displaystyle{\frac{s}{\sqrt{2}}}$",(10,10-s/(2*sqrt(2))),E);
label("$\displaystyle{\frac{s}{\sqrt{2}}}$",(10,s/(2*sqrt(2))),E);
label("$s$",(10,5),E);

draw(rightanglemark((10,0),(10,10),(0,10)));[/asy]
~~~