Efficient Tool Use with Chain-of-Abstraction Reasoning

Silin Gao[1,2*], **Jane Dwivedi-Yu**[2], **Ping Yu**[2], **Xiaoqing Ellen Tan**[2],
Ramakanth Pasunuru[2], **Olga Golovneva**[2], **Koustuv Sinha**[2]
Asli Celikyilmaz[2], **Antoine Bosselut**[1†], **Tianlu Wang**[2†]
[1]EPFL, [2]FAIR @ Meta
[1]{silin.gao,antoine.bosselut}@epfl.ch
[2]{silingao,janeyu,pingyu,ellenxtan}@meta.com
[2]{rpasunuru,olggol,koustuvs,aslic,tianluwang}@meta.com

Abstract

To achieve faithful reasoning that aligns with human expectations, large language models (LLMs) need to ground their reasoning to real-world knowledge (*e.g.*, web facts, math and physical rules). Tools help LLMs access this external knowledge, but there remains challenges for fine-tuning LLM agents (*e.g.*, Toolformer) to invoke tools in multi-step reasoning problems, where inter-connected tool calls require holistic and efficient tool usage planning.

In this work, we propose a new method for LLMs to better leverage tools in multi-step reasoning. Our method, Chain-of-Abstraction (CoA), trains LLMs to first decode reasoning chains with abstract placeholders, and then call domain tools to reify each reasoning chain by filling in specific knowledge. This planning with abstract chains enables LLMs to learn more general reasoning strategies, which are robust to shifts of domain knowledge (*e.g.*, math results) relevant to different reasoning questions. It also allows LLMs to perform decoding and calling of external tools in parallel, which avoids the inference delay caused by waiting for tool responses. In mathematical reasoning and Wiki QA domains, we show that our method consistently outperforms previous chain-of-thought and tool-augmented baselines on both in-distribution and out-of-distribution test sets, with an average $\sim 6\%$ absolute QA accuracy improvement. LLM agents trained with our method also show more efficient tool use, with inference speed being on average $\sim 1.4\times$ faster than baseline tool-augmented LLMs.

1 Introduction

Recent large language models (LLMs; Touvron et al., 2023b; Anil et al., 2023; OpenAI, 2023), have made progress at interpreting and executing instructions (Wei et al., 2021; Chung et al., 2022), but still make errors when recalling and composing world knowledge for their responses, *e.g.*, making unfactual statements (Maynez et al., 2020; Ji et al., 2023), incorrect calculations (Patel et al., 2021), etc. Using auxiliary tools (*e.g.*, a search engine to provide credible facts, a calculator for accurate math operations, etc.) at inference time can mitigate some of these errors, motivating tool-augmented language models that integrate external API calls into their output generations (Parisi et al., 2022; Schick et al., 2023; Hao et al., 2023b).

However, we show that current tool-augmented LLMs, *e.g.*, Toolformer (Schick et al., 2023), struggle to reliably and efficiently leverage tools in multi-step reasoning. In particular, tool calls in multi-step reasoning tasks are often interleaved (*i.e.*, the response of an API call is often part of the query of a subsequent call; as shown in Figure 1). Without explicitly modeling these interconnections

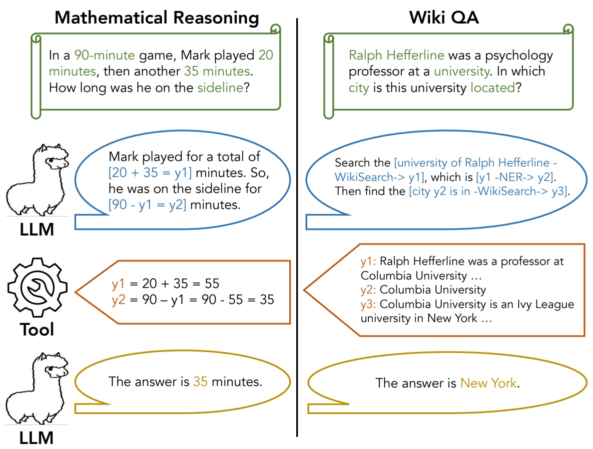

Mathematical Reasoning
In a 90-minute game, Mark played 20 minutes, then another 35 minutes. How long was he on the sideline?
LLM
Mark played for a total of [20 + 35 = y1] minutes. So, he was on the sideline for [90 - y1 = y2] minutes.
Tool
y1 = 20 + 35 = 55
y2 = 90 – y1 = 90 - 55 = 35
LLM
The answer is 35 minutes.
Wiki QA
Ralph Hefferline was a psychology professor at a university. In which city is this university located?
Search the [university of Ralph Hefferline -WikiSearch-> y1], which is [y1 -NER-> y2]. Then find the [city y2 is in -WikiSearch-> y3].
y1: Ralph Hefferline was a professor at Columbia University …
y2: Columbia University
y3: Columbia University is an Ivy League university in New York …
The answer is New York.

Figure 1: Overview of chain-of-abstraction reasoning with tools. Given a domain question (green scroll), a LLM is fine-tuned to first generate an abstract multi-step reasoning chain (blue bubble), and then call external tools to reify the chain with domain-specific knowledge (orange label). The final answer (yellow bubble) is obtained based on the reified chain of reasoning.

[*]Work done during internship at FAIR.
[†]Equal Supervision.

in reasoning chains, LLMs do not learn effective planning for tool use, which leads to less accurate reasoning with tools.[1] Meanwhile, interleaving text generation with API calls also introduces inefficient inference "waiting times," where the model must wait for the response from the API call before resuming the decoding process. This inefficiency becomes more significant in multi-step reasoning scenarios, when multiple rounds of API calls are typically required for each reasoning process.

In this work, we propose **C**hain-**o**f-**A**bstraction (**CoA**) reasoning, a robust and efficient method for LLMs to perform multi-step reasoning with tools. As shown in Figure 1, LLMs are fine-tuned with a goal of making reasoning chains with abstract placeholders. The placeholders do not affect LLMs' reasoning flow, and are subsequently infilled with specific knowledge retrieved from specialized tools, to ground the final answer generations. Planning abstract chain of reasoning encourages LLMs to inter-connect multiple tool calls and adopt more feasible reasoning strategies, which are robust to the variation of domain knowledge involved in each reasoning process, *e.g.*, specific calculation results. Unlike previous methods where LLM decoding and API calls are executed in an interleaved manner, our method leverages tools to infill knowledge **once** after the whole chain of reasoning is generated. This enables more efficient decoding across multiple examples (*e.g.*, as in a stream) because CoA traces for subsequent examples can be decoded while tool calls are made for the preceding ones, amortizing overall inference time. We develop a simple pipeline to build fine-tuning data for models to learn CoA, where we first prompt LLMs to re-write existing responses to instructions as abstract chains, and then use domain tools to check the validity of re-writing, as shown in Figure 2.

After training LLMs to learn CoA reasoning, we evaluate the finetuned models on two representative multi-step reasoning domains, including mathematical reasoning (Cobbe et al., 2021; Miao et al., 2020; Patel et al., 2021; Koncel-Kedziorski et al., 2016), and Wikipedia (Wiki) QA (Yang et al., 2018; Berant et al., 2013; Kwiatkowski et al., 2019; Joshi et al., 2017) that involves reasoning on factual descriptive knowledge. We show that our method boosts LLMs' performances, with average $\sim$7.5% and 4.5% absolute accuracy improvements on math and Wiki QA, respectively. These improvements are consistent across both in-distribution and (zero-shot) out-of-distribution test sets, and are especially pronounced on questions that require complex chain-of-thought reasoning.[2] Meanwhile, our method also uses tools more efficiently than previous augmentation methods, with average $\sim 1.47\times$ and $1.33\times$ faster inference speeds on math and Wiki QA tasks, respectively. Finally, extensive human evaluation demonstrates that our method guides LLMs to learn more accurate reasoning, which leads to $\sim$ 8% fewer reasoning errors.

2 Related Work

Tool-Augmented LLMs There is growing interest in augmenting LLMs using external tools. Considerable work has tried to adapt LLMs as tool-using reasoners through in-context learning, demonstrating promising performance improvements in various applications, *e.g.*, math problem solving (Gao et al., 2023; Chen et al., 2022), biomedical question answering (Jin et al., 2023) and self-critiquing (Gou et al., 2023). Nevertheless, guiding LLMs to effectively use tools using in-context demonstrations is challenging, which requires elaborate task-specific prompt engineering and is restricted by the model's instruction following ability (Jacovi et al., 2023). Noticing the limitations of in-context learning, several works teach LLMs to learn the usage of tools by fine-tuning (Parisi et al., 2022; Schick et al., 2023; Hao et al., 2023b), which more robustly improves LLMs' performance. However, all above approaches adopt sequential interactions with tools throughout reasoning, slowing the inference speed as a function of the latency of the tool (or API) and the number of API calls that are made.

Some other prior works focus on using LLMs for multi-step reasoning with other modules. In particular, ReAct (Yao et al., 2023b) and FireAct (Chen et al., 2023) integrate LLMs with tools into a closed loop of thought, action and observation steps. This verbose reasoning loop slows down the LLM decoding, and still incorporates tools via sequential interactions, resulting in inefficient inference. Another line of work, Program of Thoughts (Chen et al., 2022), DECLARATIVE (He-Yueya et al., 2023) and PAL (Gao et al., 2023) prompt LLMs to generate program-based reasoning and interact with code executors, which however heavily rely on closed source coding models, *i.e.*, Codex (Chen

[1]as verified by our analysis in §5

[2]*e.g.*, more than 3 steps of math derivations

et al., 2021), and are restricted to procedural arithmetic reasoning. Building on these works, CoA proposes a framework to convert natural language reasoning traces into abstract representations, and uses the abstract reasoning traces as fine-tuning data to improve tool-augmented LLMs. CoA also accelerates tool-augmented reasoning, by holistically planning the CoA traces and calling tools only once at inference time.

Tool Usage Planning Several previous works research tool usage planning in LLMs. Specifically, HuggingGPT (Shen et al., 2023), Chameleon (Lu et al., 2023), OpenAGI (Ge et al., 2023) and MetaTool (Huang et al., 2023) focus on planning the high-level sequence of using multiple tools to address multi-domain mixed tasks. Similarly, LATM (Cai et al., 2023), ML-BENCH (Liu et al., 2023) and Gorilla (Patil et al., 2023) aim at planning program-level integration of multiple APIs for designing scripts of procedural tasks, *e.g.*, a script for training a model described by a GitHub repository. ToolChain* (Zhuang et al., 2023) combines the planning of tool usage with tree-search-based reasoning (Yao et al., 2023a; Hao et al., 2023a), which is especially useful for procedural tasks (Xu et al., 2023; Cobbe et al., 2021). Different from above work, we focus on the planning of general chain-of-thought (Wei et al., 2022) reasoning with awareness of domain specialized tools.

3 Method

Chain-of-Abstraction (CoA) Reasoning Our method decouples the general reasoning of LLMs from domain-specific knowledge obtained from external tools. Figure 1 shows an overview of our method. In particular, we first fine-tune LLMs to generate reasoning chains with abstract placeholders, *e.g.*, $y1$, $y2$ and $y3$,[3] as shown in Figure 1. In the second stage, we reify each reasoning chain by replacing placeholders with domain-specific knowledge obtained from external tools, *e.g.*, calculation results from a calculator, relevant articles retrieved from web search engine, etc. Finally, the question is answered based on the reified reasoning chain.

Note that since the LLMs are trained to generate abstract chain of reasoning instead of regular chain-of-thought (CoT) reasoning with explicit values, this enables LLMs to focus on learning general and holistic reasoning strategies without needing to generate instance-specific knowledge for the model's parameters. Moreover, decoupling general reasoning and domain-specific knowledge enables LLM decoding to proceed and switch between different samples in parallel with API calling (via a pipeline), *i.e.*, LLM can start generating the next abstract chain while the tool fills the current chain, which speeds up the overall inference process.

[3]We also test placeholders in single-character format, *e.g.*, x, y and z, but these led to sub-optimal results.

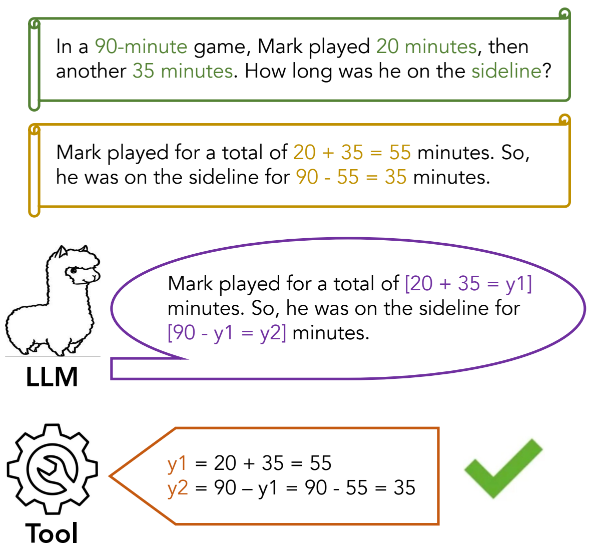

In a 90-minute game, Mark played 20 minutes, then another 35 minutes. How long was he on the sideline?
Mark played for a total of 20 + 35 = 55 minutes. So, he was on the sideline for 90 - 55 = 35 minutes.
Mark played for a total of [20 + 35 = y1] minutes. So, he was on the sideline for [90 - y1 = y2] minutes.
LLM
y1 = 20 + 35 = 55
y2 = 90 – y1 = 90 - 55 = 35
Tool

Figure 2: Illustration of gold data re-writing for fine-tuning data construction. Given a pair of domain question (green scroll) and gold answer (yellow scroll), an LLM is prompted to re-write the gold answer as a reasoning chain with abstract variables (purple bubble). Then, domain specialized tools validate the correctness of the re-writing by checking whether the abstract chain can be reified to get the final answer (orange label).

Fine-tuning Data Construction To construct chain-of-abstraction (CoA) data for fine-tuning LLMs, we collect question answering (QA) samples from existing open-source QA datasets (Cobbe et al., 2021; Miao et al., 2020; Yang et al., 2018), and prompt LLaMa-70B (Touvron et al., 2023a) to re-write the answer of each sampled question, as shown in Figure 2. Specifically, we prompt LLaMa-70B to label the spans in gold answers that correspond to knowledge operations (*e.g.*, math derivations, statements based on Wikipedia references) and then to re-write the sentences with labeled spans as fillable CoA traces, where the operation results are replaced with abstract placeholders. For example, the two derivations in the example in Figure 2 are re-written as "[$20 + 35 = y1$]" and "[$90 - y1 = y2$]", respectively.

Note that an intermediate knowledge operation result may appear multiple times in an answer, *e.g.*, in Figure 2, the first equation's result 55 is used in the second equation. We prompt LLaMa-70B to replace all occurrences of the same intermediate result with the same placeholder, thereby explicitly connecting the multiple reasoning steps. To ensure that the re-written data is accurate, we use domain-specialized tools to verify the correctness of each CoA reasoning trace.[4] Specifically, we use the tools to execute the labeled operations in each CoA, and only keep questions whose CoA can be infilled with valid results by the tools.

4 Experimental Settings

We conduct our experiments on two representative domains: mathematical reasoning and Wikipedia (Wiki) QA, which involves commonsense and logical reasoning on factual descriptive knowledge.

4.1 Mathematical Reasoning

Given a math question, the QA system needs to generate a natural language solution to the problem with step-by-step arithmetic derivations (as demonstrated in the left column of Figure 1). We assume that the derivations involved in the solution are the specialized knowledge operations required in this domain, which are labeled in square brackets with derivation results being replaced by abstract placeholders, e.g., "[$20 + 35 = y1$]".

Datasets We construct most of our fine-tuning CoA data by re-writing the GSM8K (Cobbe et al., 2021) training set, which contains 7473 linguistically diverse grade school math problems. As GSM8K dataset focuses on multi-step reasoning, it lacks coverage of single-step arithmetic problems, so we also re-write an additional set of 691 single-step math problems from the ASDiv (Miao et al., 2020) dataset. Across these re-written datasets, we find that $\sim 76.6\%$ of the CoA reasoning traces generated by LLaMa-70B are verified by our equation solver (described below). Table 1 shows the reasoning step distribution (*i.e.*, number of derivations) of our constructed fine-tuning data.

Source	Reasoning Step						
	1	2	3	4	5	>5	All
GSM8K	8	1540	1648	1164	666	553	5579
ASDiv	677	0	0	0	0	0	677

Table 1: Reasoning step distribution of correctly re-written reasoning chains in math domain.

Question	The director of the romantic comedy "Big Stone Gap" is based in what New York city?
Answer	Greenwich Village
Wikipedia References	Big Stone Gap (film) > Big Stone Gap is a 2014 American romantic comedy film directed by Adriana Trigiani. Adriana Trigiani > Adriana Trigiani is an Italian American film director based in Greenwich Village.
CoA Trace	Find the [director of romantic comedy "Big Stone Gap" -Wiki-> y1]. The name of this film's director is [y1 -NER(person)-> y2]. Then determine [y2 in what New York city -Wiki-> y3].

Table 2: Example of CoA fine-tuning data construction in Wiki QA domain.

For an in-distribution evaluation, we test models on GSM8K and ASDiv, containing 1319 and 2305 testing problems. To further test the models' generalization ability, we also conduct zero-shot evaluation on other representative math datasets, including SVAMP (Patel et al., 2021) and MAWPS (Koncel-Kedziorski et al., 2016), which contain 1000 and 2065 testing samples, respectively.[5]

Domain Tool We use an equation solver to perform the arithmetic derivations required in the math domain. Our equation solver first extracts the derivations labeled in the CoA reasoning, e.g., "[$20 + 35 = y1$]" and "[$90 - y1 = y2$]", and combines all derivations into a system of equations. Then the system of equations is solved by the SymPy toolkit,[6] to get the true value of each variable (*i.e.*, the value of the abstract placeholder). Finally, our equation solver returns the reified chain of reasoning by replacing all the variables with their solved true values (including the final answer).

4.2 Wikipedia QA

Given a question based on Wikipedia knowledge, the model needs to first identify Wikipedia articles as references related to the question, and then reason on key knowledge in the reference articles to answer the question (as shown in the right column of Figure 1). We assume that the specialized knowledge operation in this domain is the retrieval of relevant Wikipedia articles and important named-entities, which are re-written as Wikipedia searching (WikiSearch) and named-entity recognition (NER)[7] queries. Table 2 shows an example of a re-written CoA trace for Wiki QA.

[4]Detailed implementations of reasoning chain verification are described in Sec. 4.1 and 4.2.

[5]For the MAWPS benchmark, we test on the 395, 508, 562 and 600 math problems from AddSub, SingleEq, SingleOp and MultiArith portions, respectively.

[6]`https://www.sympy.org/en/index.html`

[7]We use NER to extract entities from the article that bridge the former WikiSearch results to the latter WikiSearch queries.

Datasets We use the HotpotQA (Yang et al., 2018) dataset to construct our fine-tuning CoA data in the Wiki QA domain. HotpotQA contains 113K multi-hop QA examples, each labeled with two Wikipedia articles that provide supporting knowledge. Among the 90447 training QA pairs, we identify 72991 as **Bridge** QA pairs, where an intermediate entity must be identified to link the answer to the question, as shown in Table 2. The remaining 17456 are **Comparison** QA pairs, where the attributes of two entities are compared, *e.g.*, "Are Randal Kleiser and Kyle Schickner of the same nationality?". We prompt LLaMa-70B to re-write these training QAs into CoAs with WikiSearch and NER queries, and verify each CoA with our domain tools (described below), by checking whether all the articles returned by the WikiSearch queries match one of the titles in the gold articles. Finally, 8956 Bridge QAs and 5405 Comparison QAs are used as fine-tuning data, whose re-written CoAs pass the verification.[8] For Wiki QA, we note that besides training a LLM to produce CoA data using WikiSearch, we also fine-tune a second LLM to learn to generate the final gold answer based on a correctly reified CoA reasoning trace.

We evaluate models on the HotpotQA development set, which contains 5918 Bridge QA pairs and 1487 Comparison QA pairs. Similar to the mathematical reasoning domain, we also conduct zero-shot evaluation on other open-domain QA datasets: WebQuestions (WQ; Berant et al., 2013), NaturalQuestions (NQ; Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017), which contain 2032, 3610 and 17944 test questions, respectively.

Domain Tools The specialized tools required for Wiki QA include a Wikipedia search engine to retrieve reference articles, and a NER toolkit to extract entities that bridge multi-step searching queries. We follow Toolformer (Schick et al., 2023) and implement a Wikipedia search engine as a BM25 retriever (Robertson et al., 1995; Baeza-Yates et al., 1999) that indexes the Wikipedia dump from the KILT benchmark (Petroni et al., 2021). We use the BM25 retriever to search the top-10 articles relevant to the input query, and then re-rank the articles based on their Sentence-BERT (Reimers and Gurevych, 2019) embedding cosine similarity with the question. After re-ranking, the top-1 article is selected to be the final search result.

General Class	**SpaCy NER Types included in each General Class**
person	PERSON
group	NORP, ORG, LANGUAGE
location	GPE, FAC, LOC
culture	EVENT, WORK_OF_ART, LAW, PRODUCT
date	DATE, TIME
numeral	CARDINAL, PERCENT, MONEY, QUANTITY, ORDINAL

Table 3: Aggregation of SpaCy NER types.

We use SpaCy[9] (`en_core_web_sm`) as the NER toolkit to extract named entities. To simplify NER, we aggregate the numerous SpaCy NER types into 6 general classes, as shown in Table 3. If multiple named entities are recognized, we input each recognized entity to the subsequent WikiSearch query, and select the entity whose subsequent search result has the highest Sentence-BERT embedding cosine similarity with the question.

4.3 Baselines

We apply our **CoA** reasoning method to both 7B and 70B LLaMa models, and test various model versions including the first version of LLaMa (Touvron et al., 2023a) and the more advanced LLaMa-2 and LLaMa-2-Chat (Touvron et al., 2023b). We compare our method to several baselines, including: a) few-shot prompting using 8 randomly sampled QA exemplars from the original (*i.e.*, not re-written) chain-of-thought data (**CoT-FSP**), b) fine-tuning with original chain-of-thought data (**CoT-FT**)[10], and c) **Toolformer** (Schick et al., 2023) which fine-tunes LLMs on CCNet (Wenzek et al., 2020) texts augmented with API calls. For evaluation on Wiki QA, we also compared our method with **FireAct** (Chen et al., 2023), which fine-tunes LLMs on HotpotQA ReAct (Yao et al., 2023b) trajectories distilled from GPT-4 (OpenAI, 2023).

5 Results and Analysis

5.1 Mathematical Reasoning

Table 4 shows the evaluation results for the LLaMa-2 and LLaMa-2-Chat models.[11] On the GSM8K and ASDiv datasets, our CoA method outperforms the few-shot baseline CoT-FSP and the regular fine-tuning baseline CoT-FT, demonstrating that CoA

[8]Compared to mathematical reasoning, generating CoA data for Wiki QA requires more complex tool use that combines WikiSearch and NER models, leading to a lower re-writing success rate ($\sim 15.9\%$).

[9]https://spacy.io/models/en

[10]Note that in Wiki QA domain, the HotpotQA data used for prompting or fine-tuning baselines is pre-processed to contain both gold Wikipedia articles (serving as chain-of-thought explanations) and the final answer.

[11]We include similar evaluation results for the original LLaMa model (7B) in Appendix B.

Model	Method	Use Tool	GSM8K	ASDiv	SVAMP	MAWPS				
						AddSub	SingleEQ	SingleOp	MultiArith	All
LLaMa-2 -7B	CoT-FSP	✗	16.38	47.85	38.40	52.41	63.39	82.03	43.33	60.53
	CoT-FT		35.33	57.18	48.20	66.08	74.41	85.23	65.00	73.03
	Toolformer	✓	17.59	48.55	37.10	47.34	58.46	79.54	50.67	59.81
	CoA		**37.83***	**57.61**	**51.70***	**72.15***	**82.48***	**86.48***	**73.17***	**78.89***
LLaMa-2 -Chat-7B	CoT-FSP	✗	24.03	54.14	51.30	71.90	72.44	85.41	74.00	76.32
	CoT-FT		35.41	59.00	46.90	58.23	72.24	85.41	73.00	73.37
	CoA (no Tool)		35.03	58.79	51.50	68.10	74.21	86.48	77.67	77.38
	Toolformer	✓	23.65	50.85	48.80	61.01	69.09	81.85	68.50	70.85
	Toolformer - Math		36.01	59.18	47.60	58.99	72.44	85.94	75.50	74.43
	CoA		**38.29***	**59.57**	**54.20***	**72.41**	**81.89***	**88.26***	**83.00***	**82.13***
LLaMa-2 -Chat-70B	CoT-FSP	✗	56.18	65.94	70.60	86.08	89.17	92.88	84.50	88.23
	CoT-FT		60.50	70.24	70.40	81.52	87.60	92.35	89.17	88.18
	Toolformer	✓	52.54	69.07	**73.60**	**86.84**	89.76	91.46	81.50	87.26
	Toolformer - Math		61.03	70.59	73.20	85.57	91.34	91.99	92.00	90.60
	CoA		**62.32***	**71.89***	73.40	86.33	**94.49***	**93.06**	**92.33**	**91.91***

Table 4: Evaluation results on LLaMa-2 and LLaMa-2-Chat for mathematical reasoning. "All" denotes the averaged results on four MAWPS portions. Exact match rate to the final gold answer (*i.e.*, accuracy) is reported. For each base model, the best and second-best results are **bolded** and underlined, respectively. The best results labeled with * are significantly better than their corresponding second-best results, with the significant test p-value < 0.05.

fine-tuning with tool augmentation is more effective in adapting LLMs to multi-step reasoning tasks. Similarly, when evaluated on out-of-distribution datasets, SVAMP and MAWPS, CoA also consistently outperforms the baselines. Interestingly, for these out-of-distribution datasets, CoT-FT lags further behind CoA, particularly for 7B models, showing that CoA reasoning yields more distributionally robust reasoning performance.

Our CoA method also surpasses the tool-augmented baseline Toolformer, which implies that planning the abstract variables in CoA can improve the accuracy of reasoning with tools. However, as Toolformer is not originally trained with in-domain fine-tuning data,[12] we also fine-tune a new version of Toolformer with the chain-of-thought data from GSM8K and ASDiv, denoted as **Toolformer - Math** in Table 4. We also observe that CoA performs better than Toolformer - Math, confirming that the introduction of abstract variables enables more robust tool use compared to direct integration of API calls within chain-of-thought reasoning.

Ablation Study We verify that the robust generalization performance of our CoA method does not merely benefit from using additional tools, by fine-tuning another LLM to solve the equation (from the same model backbone), rather than calling the equation solver, denoted as **CoA (no Tool)** in Table 4. We find that CoA (no Tool) performs consistently worse than CoA across all datasets, confirming that using specialized tools enables LLM agents to conduct more precise operations, rather than directly solving the same operations. However, CoA (no Tool) still outperforms all baseline methods on zero-shot generalization to SVAMP and MAWPS datasets, implying that learning abstract reasoning chains also contributes to better robustness of CoA, perhaps due to better planning of multiple reasoning steps indexed by abstract variables.

Reasoning Steps Our findings suggest that the benefits of chain-of-abstraction reasoning are most pronounced when problems require long reasoning chains to be solved. Figure 3 shows the stratified performance of three models on GSM8K QA, relative to the number of reasoning steps in the predicted and gold reasoning chains. Compared to the few-shot CoT-FSP, CoA produces reasoning chains that more often match the length of the gold reasoning chains, as reflected by the heat-map statistics (left column) being more aggregated around the diagonal (comparable to CoT-FT). At the same time, we observe that models achieve better QA accuracy when the number of reasoning steps in their generated answers are aligned with the gold references (*i.e.*, the diagonal of heat-maps in right column). Above results show that fine-tuned models are better at learning to produce reasoning chains that match the true reasoning chain for the problem.

[12] Toolformer is fine-tuned on CCNet data, which may not contain rich mathematical reasoning samples.

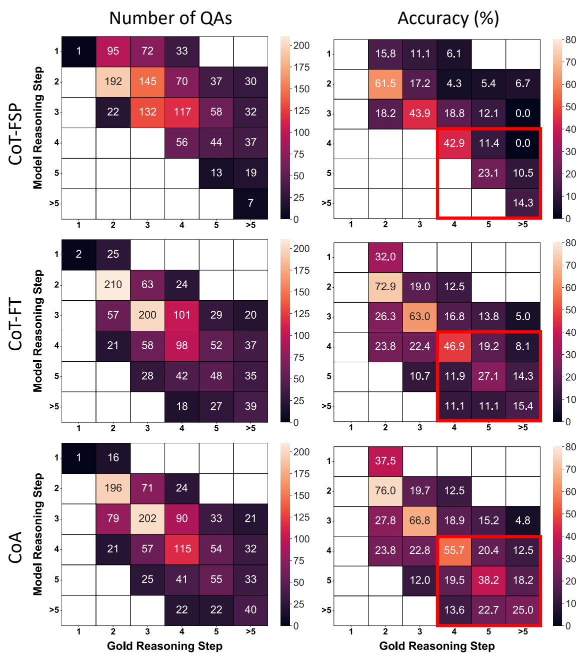


Figure 3: GSM8K evaluation results on LLaMa-2-Chat-7B *w.r.t.* the number of reasoning steps in the predicted and gold reasoning chain. (Left) The number of test examples that belong to each stratum. (Right) The corresponding model accuracy (%) for those examples. Non-diagonal cells with fewer than 15 examples are ignored.

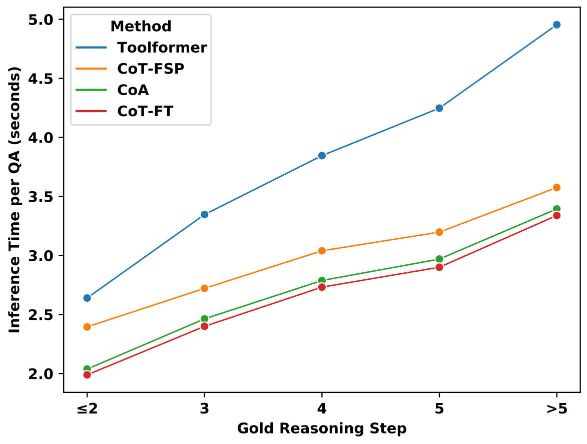


Figure 4: Wall-clock inference time on GSM8K (seeded with LLaMa-2-Chat-7B). Average time of answering a question is measured (in seconds) *w.r.t.* the number of gold reasoning steps required for the question.

Method	Accuracy
CoT-FSP	27.90
CoT-FT	39.12
Toolformer	24.56
Toolformer - Math	35.25
CoA	**40.79**

Table 6: Evaluation results on GSM8K with self-consistency decoding (seeded with LLaMa-2-Chat-7B). Each model uses majority voting to aggregate the answers of 16 sampled reasoning chains

Method	Error Rate	
	Arithmetic	Reasoning
CoT-FSP	17.3	70.3
CoT-FT	25.2	67.8
CoA	**0.0**	**60.4**

Table 5: Human evaluation results of arithmetic and reasoning error rates on 200 GSM8K test samples. Models developed based on LLaMa-2-Chat-7B are presented.

Interestingly, we find that CoA, compared to CoT-FT, achieves higher performance especially on questions that require more reasoning steps. In the right column of Figure 3, CoA's improvement over CoT-FT is more pronounced on questions with more than 3 steps in the gold reasoning chain (highlighted with red squares). This indicates that the model trained with CoA has more robust long chain-of-thought reasoning capability, which is learned from planning with abstractions.

Human Evaluation To more comprehensively verify that CoA improves both knowledge operation (*i.e.*, arithmetic by using tools) and reasoning accuracy, we conduct a human evaluation on different model answers to 200 randomly sampled GSM8K test questions. Specifically, given a GSM8K question and a model's answer to the question, we ask human workers to judge whether the answer contains any arithmetic errors (*e.g.*, wrong calculations, invalid equations) or reasoning errors unrelated to math derivations (*e.g.*, misunderstanding of the question, improper strategy for solving the question), and report how often the model makes these two kinds of errors. In Table 5, we find that CoA effectively reduces arithmetic errors to zero, due to the use of equation solver to perform accurate calculations. More importantly, our method also makes fewer reasoning errors compared to the baselines, verifying that CoA fine-tuning guides the model to learn more accurate reasoning through the holistic planning of abstract reasoning chains. By contrast, ordinary fine-tuning (*i.e.*, CoT-FT) produces a more limited reasoning improvement compared to the few-shot CoT-FSP, while also failing to suppress arithmetic errors.

Inference Efficiency Importantly, we find that the performance benefits of CoA reasoning do not come with increased computational costs. In Fig-

Model	Method	Use Tool	HotpotQA				WQ	NQ	TriviaQA
			Bridge	Comparison	Both	Time			
LLaMa-2 -Chat-7B	CoT-FSP	✗	11.69	45.46	18.47	2.074	34.65	30.91	53.48
	CoT-FT	✗	14.24	<u>56.69</u>	22.77	<u>1.937</u>	33.51	25.40	51.05
	Toolformer	✓	12.99	44.59	20.00	2.350	<u>36.22</u>	30.22	54.15
	Toolformer - Wiki	✓	15.68	56.42	23.86	2.301	**36.61**	32.96	<u>55.08</u>
	FireAct	✓	<u>19.18</u>	54.14	<u>26.20</u>	2.706	36.02	<u>35.87</u>	52.96
	CoA	✓	**21.00***	**56.96**	**28.22***	**1.896**	35.97	**38.67***	**57.90***
LLaMa-2 -Chat-70B	CoT-FSP	✗	21.39	56.62	28.47	6.668	34.89	37.42	63.61
	CoT-FT	✗	23.84	<u>63.95</u>	31.90	<u>6.401</u>	34.15	39.75	62.28
	Toolformer	✓	22.24	56.09	29.04	6.888	<u>37.16</u>	40.42	64.31
	Toolformer - Wiki	✓	<u>26.38</u>	63.82	<u>33.90</u>	6.855	**37.70**	<u>41.25</u>	<u>66.64</u>
	CoA	✓	**27.61***	**64.09**	**34.94***	**6.369**	36.37	**43.57***	**69.08***

Table 7: Wiki QA evaluation results on LLaMa-2-Chat-based models. "Both" denotes the overall evaluation results on both bridge and comparison portions of HotpotQA. "Time" denotes the average seconds that each agent needs to answer a question in HotpotQA. Exact match rate to the final gold answer (*i.e.*, accuracy) is reported. For each base model, the best and second-best results are **bolded** and <u>underlined</u>, respectively. The best results labeled with * are significantly better than their corresponding second-best results, with the significant test p-value < 0.05.

ure 4, we show the average time (seconds) that CoA and baseline agents (seeded with LLaMa-2-Chat-7B) needs to answer a question *w.r.t.* required gold reasoning steps. Compared to the CoT baselines, CoA requires less time than the few-shot baseline CoT-FSP, whose generation needs to be conditioned on additional examples. However, CoA is slightly less inference-efficient compared to CoT-FT, likely due to the decoding of additional tokens (*e.g.*, "[" and "]") for the abstract statements.

Compared to Toolformer, CoA has a lower and flatter inference time curve, indicating better scaling as the number of reasoning steps increases. This difference arises because CoA decouples the generation of (abstract) reasoning chains from the retrieval of knowledge (*i.e.*, tool use), allowing full reasoning chains to be decoded before any tool is called. This procedure amortizes inference costs in two ways. First, tool calls are made after the CoA trace has been decoded, enabling parallel tool calls for the same trace (*e.g.*, using an equation solver once rather than multiple calls to a calculator), and avoiding the time delay caused by waiting for external API responses. Consequently, the model fine-tuned with CoA is more efficient at multi-step reasoning, especially when the number of reasoning steps (*i.e.*, tool calls) increases. Second, across multiple examples, the model can generate the CoA trace of the next example while tool calls are made for the preceding one, parallelizing CoA decoding and tools calls across examples.

Self-Consistency Decoding Besides of greedy decoding, we also test more advanced inference strategy, *i.e.*, self-consistency (Wang et al., 2022) decoding, on our CoA reasoning method. We test all methods on the GSM8K dataset seeded with LLaMa-2-Chat-7B. Each method samples 16 reasoning chains and uses majority voting to aggregate the 16 answers derived by the reasoning chains, to get the final answer. For the hyperparameters of sampling, we set the temperature, top-k and top-p as 1.0, 40 and 0.5, respectively. Table 6 shows our evaluation results. We find that our CoA method consistently outperforms all baseline methods when shifting from greedy decoding to self-consistency decoding. This shows that our method also has better potential to be generalized to different LLM decoding schemes.

5.2 Wiki QA

Table 7 shows our Wiki QA results using LLaMa-2-Chat models.[13] Similar to mathematical reasoning, we fine-tune a new version of Toolformer with in-domain chain-of-thought data from HotpotQA, denoted as **Toolformer - Wiki**. On HotpotQA, CoA achieves higher exact match rates with the gold reference compared to the few-shot or fine-tuning baselines. In particular, CoA outperforms all baselines on the more challenging bridge-type QAs, where two steps of reasoning over Wikipedia knowledge are *consecutively* entangled, *i.e.*, can-

[13]We include similar evaluation results on LLaMa-2-7B in Appendix B.

not be performed independently in parallel as in comparison-type QAs. Compared to FireAct fine-tuning, CoA also achieves better performance on both bridge and comparison QAs, without requiring data distilled from closed source GPT-4.

As with mathematical reasoning, CoA agents also perform more efficient inference than Toolformer and FireAct agents when answering HotpotQA questions. We also find that CoA is more efficient (**Time** column) than both CoT-FSP and CoT-FT, as CoA does not require few-shot examples as additional inputs and does not need to generate long Wiki articles, which are instead provided by the search engine. Finally, CoA improves over the baseline methods in zero-shot generalization experiments on other Wiki QA datasets, outperforming all baselines on NaturalQuestions and TriviaQA, and matching the best baselines on WebQuestions.

6 Conclusion

In this work, we propose to decouple the general reasoning of LLM agents from specialized knowledge obtained via external tools. Our method, chain-of-abstraction (CoA), encourages LLMs to learn the planning of abstract multi-step reasoning, which are more robust to out-of-distribution knowledge shifts. CoA also achieves a more efficient pipeline for tool usage that significantly improves the speed of tool-augmented multi-step reasoning. The simple, yet effective, implementations of our method on two diverse tasks (*i.e.*, math reasoning and open-domain QA) demonstrate its potential for being adapted to new reasoning scenarios.

Limitations

We acknowledge a few limitations in our work. First, datasets used for testing our method cannot have exhaustive coverage of all real-world reasoning scenarios. We instead consider two representative reasoning domains, *i.e.*, mathematical reasoning and general open-domain (Wikipedia) QA, and use English as a primary language in our testing. Furthermore, our method is tested on the setting of fine-tuning the full LLMs, which requires considerable computational resources, while more efficient model training schemes, *e.g.*, LoRA (Hu et al., 2021), can be applied in future work.

Acknowledgements

We thank Beatriz Borges, Gail Weiss, Syrielle Montariol, Li Mi and Zeming Chen for reading and providing comments on drafts of this paper. Antoine Bosselut gratefully acknowledges the support of the Swiss National Science Foundation (No. 215390), Innosuisse (PFFS-21-29), the EPFL Science Seed Fund, the EPFL Center for Imaging, Sony Group Corporation, and the Allen Institute for AI.

References

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

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*, volume 463. ACM press New York.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2023. Large language models as tool makers. *arXiv preprint arXiv:2305.17126*.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv: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.

Yingqiang Ge, Wenyue Hua, Jianchao Ji, Juntao Tan, Shuyuan Xu, and Yongfeng Zhang. 2023. Openagi: When llm meets domain experts. *arXiv preprint arXiv:2304.04370*.

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

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023a. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023b. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *arXiv preprint arXiv:2305.11554*.

Joy He-Yueya, Gabriel Poesia, Rose E Wang, and Noah D Goodman. 2023. Solving math word problems by combining language models with symbolic solvers. *arXiv preprint arXiv:2304.09102*.

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*.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, et al. 2023. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv preprint arXiv:2310.03128*.

Alon Jacovi, Avi Caciularu, Jonathan Herzig, Roee Aharoni, Bernd Bohnet, and Mor Geva. 2023. A comprehensive evaluation of tool-assisted generation strategies. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13856–13878.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55:1–38.

Qiao Jin, Yifan Yang, Qingyu Chen, and Zhiyong Lu. 2023. Genegpt: Augmenting large language models with domain tools for improved access to biomedical information. *Preprint*, arXiv:2304.09667.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

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.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Yuliang Liu, Xiangru Tang, Zefan Cai, Junjie Lu, Yichi Zhang, Yanjun Shao, Zexuan Deng, Helan Hu, Zengxian Yang, Kaikai An, et al. 2023. Mlbench: Large language models leverage open-source libraries for machine learning tasks. *arXiv preprint arXiv:2311.09835*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. *arXiv preprint arXiv:2304.09842*.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. 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*, pages 975–984.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

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.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. 2021. Kilt: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

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*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

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.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Édouard Grave. 2020. Ccnet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. 2023. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A Rossi, Somdeb Sarkhel, and Chao Zhang. 2023. Toolchain*: Efficient action space navigation in large language models with a* search. *arXiv preprint arXiv:2310.13227*.

A Implementation Details

Evaluation Details For mathematical reasoning evaluation, we extract the last number appeared in each model's answer, and check whether the number exactly match the gold reference. The accuracy is reported as the rate of such exact match across all QAs in a test set. For Wiki QA evaluation, similar to mathematical reasoning, we extract the final answer of each model and calculate its exact match rate to the gold reference. Specifically, the final answer is supposed to be the words after "Action: finish[" for FireAct baseline, and words after "The answer is " for other models. Our 8-shot in-domain examples used for the CoT-FSP baseline are shown in Table 14 and 15, which enables the model to provide answer with our required format for evaluation, *i.e.*, stating its final answer after "The answer is ". Our human evaluation on GSM8K is conducted by 5 internal domain experts from our research group. For each math question, we provide the experts with the gold answer as reference, and ask them to evaluate each model answer in anonymous manner, *i.e.*, experts do not know which model each answer comes from. Two yes-or-no questions are asked

Model	Method	GSM8K	ASDiv	SVAMP	MAWPS				
					AddSub	SingleEQ	SingleOp	MultiArith	All
LLaMa-7B	CoT-FSP	11.90	44.69	31.80	56.20	59.65	70.28	43.00	57.05
	CoT-FT	30.71	53.19	42.30	55.70	69.09	77.05	54.17	64.36
	CoA	**35.71**	**56.36**	**51.10**	**67.59**	**80.51**	**85.94**	**68.33**	**75.98**
LLaMa-2-7B	CoT-FSP	16.38	47.85	38.40	52.41	63.39	82.03	43.33	60.53
	CoT-FT	35.33	57.18	48.20	66.08	74.41	85.23	65.00	73.03
	Toolformer	17.59	48.55	37.10	47.34	58.46	79.54	50.67	59.81
	CoA	**37.83**	**57.61**	**51.70**	**72.15**	**82.48**	**86.48**	73.17	**78.89**
LLaMa-2-Chat-7B	CoT-FSP	24.03	54.14	51.30	71.90	72.44	85.41	74.00	76.32
	CoT-FT	35.41	59.00	46.90	58.23	72.24	85.41	73.00	73.37
	CoT-FT (no ASDiv)	36.19	44.93	35.30	38.48	52.95	61.21	77.67	59.61
	Toolformer	23.65	50.85	48.80	61.01	69.09	81.85	68.50	70.85
	Toolformer - Math	36.01	59.18	47.60	58.99	72.44	85.94	75.50	74.43
	CoA	38.29	**59.57**	**54.20**	**72.41**	**81.89**	**88.26**	83.00	**82.13**
	CoA (no ASDiv)	**39.73**	54.19	44.40	54.18	73.62	73.49	**85.33**	73.27
	CoA (no Tool)	35.03	58.79	51.50	68.10	74.21	86.48	77.67	77.38
LLaMa-2-Chat-70B	CoT-FSP	56.18	65.94	70.60	86.08	89.17	92.88	84.50	88.23
	CoT-FT	60.50	70.24	70.40	81.52	87.60	92.35	89.17	88.18
	Toolformer	52.54	69.07	**73.60**	**86.84**	89.76	91.46	81.50	87.26
	Toolformer - Math	61.03	70.59	73.20	85.57	91.34	91.99	92.00	90.60
	CoA	**62.32**	**71.89**	73.40	86.33	**94.49**	**93.06**	**92.33**	**91.91**
GPT-J	Toolformer	-	40.4	29.4	-	-	-	-	44.0

Table 8: Mathematical reasoning evaluation results.

for evaluating each model answer, including: a) whether the answer has any arithmetic error, and b) whether the answer has any reasoning error, and binary choices from the experts are collected to calculate the error rates of each model's generation. We present our detailed instructions for human evaluation in Figure 5. Our data collection protocol is approved by our organization in terms of ethics.

Model Training We fine-tune our models with batch size 8 and learning rate $2e^{-5}$ and $1e^{-5}$ for 7B and 70B model sizes, respectively, using cosine learning rate scheduler with warm-up step 10. We use AdamW (Loshchilov and Hutter, 2018) optimizer for all our fine-tuning experiments, with β_1, β_2 and ϵ set to 0.9, 0.95 and $1e^{-8}$, respectively. Training weight decay is set to 0.1. For mathematical reasoning, we use a total of 400 training steps, and get the best model checkpoints (with highest validation scores) at step 240 and 200 for 7B and 70B model sizes. For Wiki QA domain, we adjust the total training steps to 500, and get the best checkpoints at step 450 and 300 for 7B and 70B models. Therefore, only $\sim$2K and $\sim$3K QAs are required in practice for fine-tuning our models in math and Wiki QA domains. The training of our 7B and 70B models is based on 8 and 64 NVIDIA A100-SXM4 (80GB) GPUs, with training time about 2 and 5 hours per model, respectively.

B Full Experimental Results

Table 8 and 9 show the full results of our experiments on math and Wiki QA domains. Our method of CoA achieves consistent improvements over baselines across various LLaMa model versions (LLaMa, LLaMa-2 and LLaMa-2-Chat), model sizes (7B and 70B), and domain benchmarks. This shows great potential of our method being generalized to new model backbones and reasoning tasks. We also present results on GSM8K subsets according to varying numbers of gold reasoning steps in Table 10, where we confirm that CoA has more robust long chain-of-thought reasoning accuracy.

Fine-Tuning Data Balance In the mathematical reasoning domain, we also validate the importance of using fine-tuning data that is balanced across different reasoning steps. Specifically, we conduct an ablation study on CoT-FT and CoA seeded with LLaMa-2-Chat-7B model, by removing the single-step QA samples of ASDiv from the fine-tuning data (**no ASDiv**). We find that CoT-FT (no ASDiv) and CoA (no ASDiv) turn out to be biased towards multi-step reasoning, where they achieve better performance on GSM8K and MultiArith that contain mainly multi-step QAs, but suffer from severe performance degradation on other datasets that contain many single-step math problems. This demonstrates that maintaining a good balance of single-step and multi-step reasoning data is impor-

Model	Method	HotpotQA			WebQ.	NaturalQ.	TriviaQA
		Bridge	Comparison	All			
LLaMa-2-7B	CoT-FSP	14.43	45.26	20.62	33.96	33.35	56.95
	CoT-FT	14.85	57.36	23.39	31.50	26.93	52.32
	Toolformer	14.12	42.76	20.35	**37.11**	34.49	57.79
	CoA	**22.00**	**57.43**	**29.12**	34.60	**38.28**	**58.28**
LLaMa-2-Chat-7B	CoT-FSP	11.69	45.46	18.47	34.65	30.91	53.48
	CoT-FT	14.24	56.69	22.77	33.51	25.40	51.05
	Toolformer	12.99	44.59	20.00	36.22	30.22	54.15
	Toolformer - Wiki	15.68	56.42	23.86	**36.61**	32.96	55.08
	FireAct	19.18	54.14	26.20	36.02	35.87	52.96
	CoA	**21.00**	**56.96**	**28.22**	35.97	**38.67**	**57.90**
LLaMa-2-Chat-70B	CoT-FSP	21.39	56.62	28.47	34.89	37.42	63.61
	CoT-FT	23.84	63.95	31.90	34.15	39.75	62.28
	Toolformer	22.24	56.09	29.04	37.16	40.42	64.31
	Toolformer - Wiki	26.38	63.82	33.90	**37.70**	41.25	66.64
	CoA	**27.61**	**64.09**	**34.94**	36.37	**43.57**	**69.08**
GPT-J	Toolformer	-	-	-	26.3	17.7	48.8

Table 9: Wiki QA evaluation results.

tant for adapting LLMs to be robust reasoners.

More Prompting Baselines We also compare our CoA reasoning method to more prompting-based methods PAL (Gao et al., 2023) and DECLARATIVE (He-Yueya et al., 2023), which use few-shot coding demonstrations to prompt math solutions as Python or declarative programs. Table 11 shows our comparison results on the GSM8K dataset, where all methods are seeded with LLaMa-2-Chat-7B. Without seeding with dedicated coding models (*e.g.*, code-davinci-002), PAL and DECLARATIVE get far lower accuracy on GSM8K, which significantly under-perform our CoA method, and even ordinary CoT-FSP.

In contrast, our CoA method relies less on artificial demonstrations and distributional closeness of the seed LLM to target tasks, as CoA fine-tunes the LLM agent on pre-defined abstract reasoning chains, acquired from simple rewriting of natural language reasoning traces. Consequently, CoA is flexible in various generation formats, *e.g.*, code and plain text, and generalizes well from mathematical reasoning to open-domain QA, which is a very different type of reasoning task. This indicates our method's generalizability to novel reasoning schemes required by a new domain.

C Fine-Tuning Data Re-writing Details

Table 12 and 13 show the prompting examples for fine-tuning data construction of our method. We prompt LLaMa-70B to re-write existing math and Wiki QAs as abstract reasoning chains, which gets rid of data distillation from close-sourced LLMs, yet obtains data resources that enable more effective learning of multi-step reasoning.

Method	Gold Reasoning Step				
	≤ 2	3	4	5	> 5
CoT-FSP	42.9	26.3	18.0	10.9	3.6
CoT-FT	55.5	42.6	25.8	19.0	10.8
CoA	**55.8**	**44.4**	**32.5**	**25.3**	**15.1**
	+0.3	+1.8	+6.7	+6.3	+4.3

Table 10: Stratified LLaMa-2-Chat-7B evaluation results on GSM8K with different gold reasoning steps. The last row reports absolute accuracy improvement of our CoA method compared to CoT-FT baseline.

Method	Accuracy
CoT-FSP	24.03
PAL	20.55
DECLARATIVE	9.86
CoA	**38.29**

Table 11: Comparison of CoA to prompting-based methods on GSM8K, seeded with LLaMa-2-Chat-7B.

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees will the grove workers plant today?
A: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21-15=6. The answer is 6.
C: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been [21 - 15 = y1]. The answer is y1.

Q: The flowers cost $9, the clay pot costs $20 more than the flower, and the bag of soil costs $2 less than the flower. How much does it cost to plant the flowers?
A: The clay pot costs $20 + $9 = $29. The bag of soil costs $9 - $2 = $7. The cost to plant the flowers is $9 + $29 + $7 = $45. The answer is 45.
C: The clay pot costs [20 + 9 = y1]. The bag of soil costs [9 - 2 = y2]. The cost to plant the flowers is [9 + y1 + y2 = y3]. The answer is y3.

Q: From March to August, Sam made $460 doing 23 hours of yard work. However, from September to February, Sam was only able to work for 8 hours. If Sam is saving up to buy a video game console that costs $600 and has already spent $340 to fix his car, how many more hours does he need to work before he can buy the video game console?
A: Sam makes $460 / 23 hrs = $20/hr. From September to February, Sam made 8hrs x $20/hr = $160. From March to February, Sam made a total of $460 + $160 = $620. After fixing his car, he was left with $620 - $340 = $280. Sam needs another $600 - $280 = $320. Sam needs to work another $320 / $20/hr = 16 hours. The answer is 16.
C: Sam makes [460 / 23 = y1] dollars per hour. From September to February, Sam made [8 * y1 = y2] dollars. From March to February, Sam made a total of [460 + y2 = y3] dollars. After fixing his car, he was left with [y3 - 340 = y4]. Sam needs another [600 - y4 = y5] dollars. Sam needs to work another [y5 / y1 = y6] hours. The answer is y6.

Q: There were nine computers in the server room. Five more computers were installed each day, from Monday to Thursday. How many computers are now in the server room?
A: There were originally 9 computers. For each of 4 days, 5 more computers were added. So 5 * 4 = 20 computers were added. 9 + 20 is 29. The answer is 29.
C: There were originally 9 computers. For each of 4 days, 5 more computers were added. So [5 * 4 = y1] computers were added. [9 + y1 = y2]. The answer is y2.

Q: Of the 90 people on William's bus, 3/5 were Dutch. Of the 1/2 of the Dutch who were also American, 1/3 got window seats. What's the number of Dutch Americans who sat at the windows?
A: On the bus, the number of Dutch people was 3/5 of the total number, a total of 3/5 x 90 = 54 people. Out of the 54 people who were Dutch, 1/2 were Dutch Americans, a total of 1/2 x 54 = 27 people. If 1/3 of the passengers on the bus identifying as Dutch Americans sat at the windows, their number is 1/3 x 27 = 9. The answer is 9.
C: On the bus, the number of Dutch people was 3/5 of the total number, a total of [3/5 * 90 = y1] people. Out of the Dutch people, 1/2 were Dutch Americans, a total of [1/2 * y1 = y2] people. If 1/3 of the passengers on the bus identifying as Dutch Americans sat at the windows, their number is [1/3 * y2 = y3]. The answer is y3.

Table 12: Prompting examples for fine-tuning data construction in mathematical reasoning domain. Given a question (Q) and a gold answer (A), LLaMa-70B is prompted to generate the re-writing of answer as abstract reasoning chain (C). Based on that, our method trains a LLM to generate the abstract chain based on the question, and the final answer is derived by reify the chain of reasoning with the domain tool (*i.e.*, equation solver).

Q: Fritz von Brodowski was killed during what global war that lasted from 1939 to 1945?
A: The answer is World War II.
W: Fritz von Brodowski > Friedrich Wilhelm Konrad von Brodowski was controversially killed while in French custody during World War II.
C: Find the [war in which Fritz von Brodowski was killed -Wiki-> y1].

Q: Which tennis player won more Grand Slam titles, Henri Leconte or Jonathan Stark?
A: The answer is Jonathan Stark.
W: Henri Leconte > He won the French Open men's doubles title in 1984. Jonathan Stark (tennis) > During his career he won two Grand Slam doubles titles.
C: First identify the [number of Grand Slam titles Henri Leconte won -Wiki-> y1]. Then find out the [number of Grand Slam titles Jonathan Stark won -Wiki-> y2].

Q: The director of the romantic comedy "Big Stone Gap" is based in what New York city?
A: The answer is Greenwich Village.
W: Big Stone Gap (film) > Big Stone Gap is a 2014 American romantic comedy film directed by Adriana Trigiani. Adriana Trigiani > Adriana Trigiani is an Italian American film director based in Greenwich Village.
C: First search the [director of romantic comedy "Big Stone Gap" -Wiki-> y1]. The name of this film's director is [y1 -NER(person)-> y2]. Then determine [y2 in what New York city -Wiki-> y3].

Q: Are Randal Kleiser and Kyle Schickner of the same nationality?
A: The answer is yes.
W: Randal Kleiser > John Randal Kleiser (born July 20, 1946) is an American film director and producer. Kyle Schickner > Kyle Schickner is an American film producer, writer, director, actor.
C: First find out the [nationality of Randal Kleiser -Wiki-> y1]. Then figure out the [nationality of Kyle Schickner -Wiki-> y2].

Q: Extras was created, written, and directed by Ricky Dene Gervais, an English comedian, actor, writer, producer, director, singer, and musician, born on which date?
A: The answer is 25 June 1961.
W: Ricky Gervais > Ricky Dene Gervais (born 25 June 1961) is an English comedian, actor, writer, producer, director, singer, and musician.
C: Search [when Ricky Dene Gervais was born -Wiki-> y1].

Q: Sameera Perera is a cricketer from what island country located southeast of the Republic of India and northeast of the Maldives?
A: The answer is Sri Lanka.
W: Sameera Perera > Sameera Perera (born 20 August 1988) is a Sri Lankan cricketer.
C: Identify the [country that cricketer Sameera Perera is from -Wiki-> y1].

Q: What screenwriter with credits for "Evolution" co-wrote a film starring Nicolas Cage and Téa Leoni?
A: The answer is David Weissman.
W: The Family Man > The Family Man is a 2000 American romantic comedy-drama film starring Nicolas Cage and Téa Leoni. David Weissman > His film credits include "The Family Man" (2000), "Evolution" (2001), and "When in Rome" (2010).
C: First figure out the [film of Nicolas Cage and Téa Leoni -Wiki-> y1]. The name of this film is [y1 -NER(culture)-> y2]. Then find out [who wrote y2 with credits for "Evolution" -Wiki-> y3].

Q: Ralph Hefferline was a psychology professor at a university that is located in what city?
A: The answer is New York City.
W: Ralph Hefferline > Ralph Franklin Hefferline was a psychology professor at Columbia University. Columbia University > Columbia University is a private Ivy League research university in Upper Manhattan, New York City.
C: First identify the [university of psychology professor Ralph Hefferline -Wiki-> y1]. The university of this professor is [y1 -NER(group)-> y2]. Then figure out [y2 is in what city -Wiki-> y3].

Table 13: Prompting examples for fine-tuning data construction in Wiki QA domain. Given a question (Q), a gold answer (A) and its supporting Wikipedia articles (W), LLaMa-70B is prompted to generate an abstract reasoning chain (C) with Wikipedia searching and NER queries. Based on that, our method first trains a LLM to generate the abstract chain of queries based on the question, and then execute the queries by domain tools (*i.e.*, Wikipedia search engine and NER toolkit). Finally, a second LLM is trained to generate the final answer based on the Wikipedia searching results (excluding intermediate NER results) in the reified chain of reasoning.

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees will the grove workers plant today?
A: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21-15=6. The answer is 6.

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
A: There are originally 3 cars. 2 more cars arrive. 3 + 2 = 5. The answer is 5.

Q: The flowers cost $9, the clay pot costs $20 more than the flower, and the bag of soil costs $2 less than the flower. How much does it cost to plant the flowers?
A: The clay pot costs $20 + $9 = $29. The bag of soil costs $9 - $2 = $7. The cost to plant the flowers is $9 + $29 + $7 = $45. The answer is 45.

Q: Maddie wants to see how much her mom spends on coffee each week. She makes herself 2 cups of coffee per day. Each cup has 1.5 ounces of coffee beans. A bag of coffee costs $8 and contains 10.5 ounces of beans. How much does she spend on her coffee per week?
A: She uses 3 ounces of beans per day because 2 x 1.5 = 3. She uses 21 ounces of beans per week because 7 x 3 = 21. She buys 2 bags of beans per week because 21 / 10.5 = 2. She spends $16 on the beans per week because 2 x 8 = 16. The answer is 16.

Q: There were nine computers in the server room. Five more computers were installed each day, from Monday to Thursday. How many computers are now in the server room?
A: There were originally 9 computers. For each of 4 days, 5 more computers were added. So 5 * 4 = 20 computers were added. 9 + 20 is 29. The answer is 29.

Q: From March to August, Sam made $460 doing 23 hours of yard work. However, from September to February, Sam was only able to work for 8 hours. If Sam is saving up to buy a video game console that costs $600 and has already spent $340 to fix his car, how many more hours does he need to work before he can buy the video game console?
A: Sam makes $460 / 23 hrs = $20/hr. From September to February, Sam made 8hrs x $20/hr = $160. From March to February, Sam made a total of $460 + $160 = $620. After fixing his car, he was left with $620 - $340 = $280. Sam needs another $600 - $280 = $320. Sam needs to work another $320 / $20/hr = 16 hours. The answer is 16.

Q: Of the 90 people on William's bus, 3/5 were Dutch. Of the 1/2 of the Dutch who were also American, 1/3 got window seats. What's the number of Dutch Americans who sat at the windows?
A: On the bus, the number of Dutch people was 3/5 of the total number, a total of 3/5 x 90 = 54 people. Out of the 54 people who were Dutch, 1/2 were Dutch Americans, a total of 1/2 x 54 = 27 people. If 1/3 of the passengers on the bus identifying as Dutch Americans sat at the windows, their number is 1/3 x 27 = 9. The answer is 9.

Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
A: Originally, Leah had 32 chocolates. Her sister had 42. So in total they had 32 + 42 = 74. After eating 35, they had 74-35=39. The answer is 39.

Table 14: Few-shot examples used for CoT-FSP baseline model in mathematical reasoning domain.

Q: Fritz von Brodowski was killed during what global war that lasted from 1939 to 1945?
A: Fritz von Brodowski > Friedrich Wilhelm Konrad von Brodowski was controversially killed while in French custody during World War II. The answer is World War II.

Q: Which tennis player won more Grand Slam titles, Henri Leconte or Jonathan Stark?
A: Henri Leconte > He won the French Open men's doubles title in 1984. Jonathan Stark (tennis) > During his career he won two Grand Slam doubles titles. The answer is Jonathan Stark.

Q: The director of the romantic comedy "Big Stone Gap" is based in what New York city?
A: Big Stone Gap (film) > Big Stone Gap is a 2014 American romantic comedy film directed by Adriana Trigiani. Adriana Trigiani > Adriana Trigiani is an Italian American film director based in Greenwich Village. The answer is Greenwich Village.

Q: Are Randal Kleiser and Kyle Schickner of the same nationality?
A: Randal Kleiser > John Randal Kleiser (born July 20, 1946) is an American film director and producer. Kyle Schickner > Kyle Schickner is an American film producer, writer, director, actor. The answer is yes.

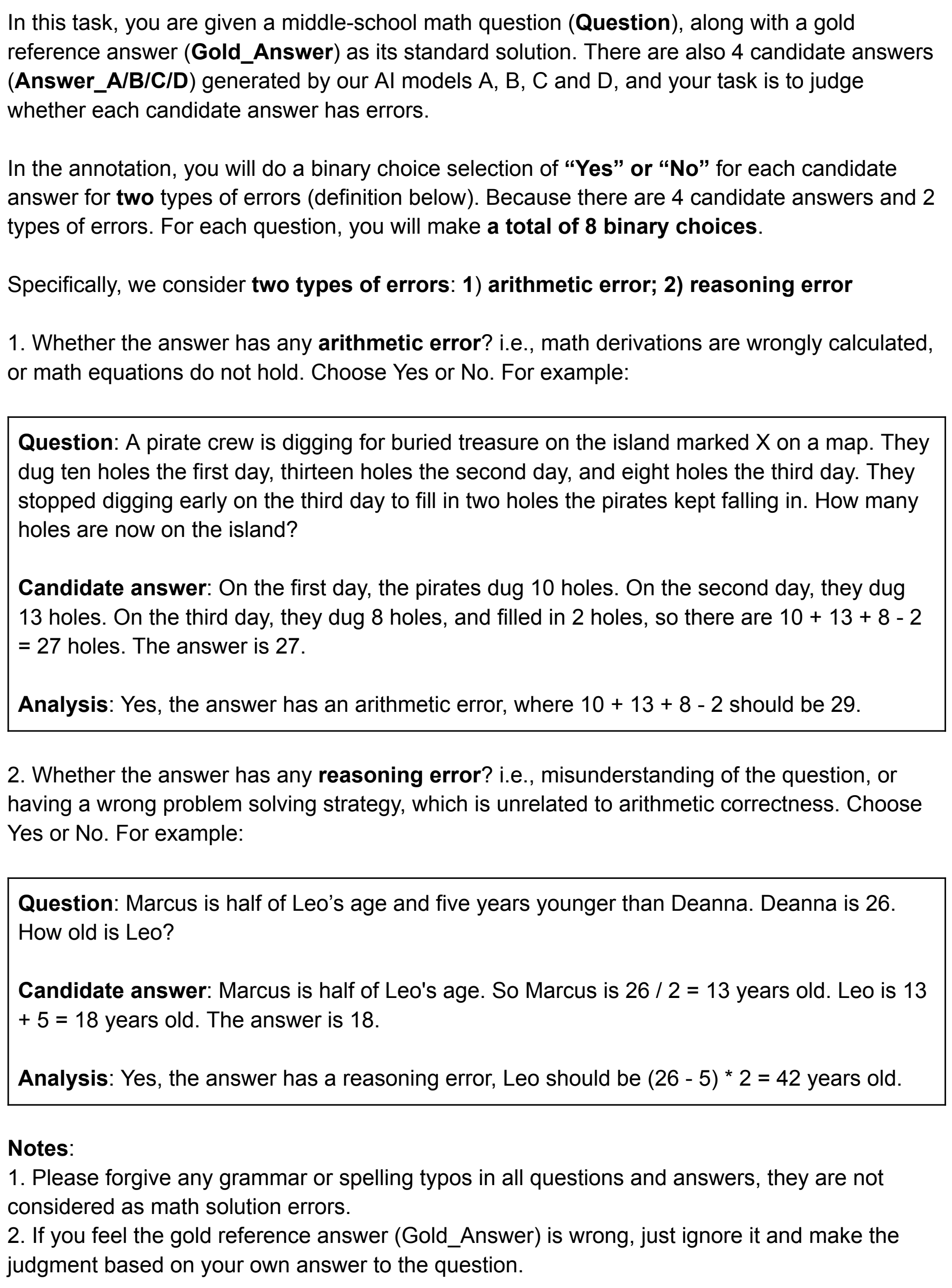
Q: Extras was created, written, and directed by Ricky Dene Gervais, an English comedian, actor, writer, producer, director, singer, and musician, born on which date?
A: Ricky Gervais > Ricky Dene Gervais (born 25 June 1961) is an English comedian, actor, writer, producer, director, singer, and musician. The answer is 25 June 1961.

Q: Sameera Perera is a cricketer from what island country located southeast of the Republic of India and northeast of the Maldives?
A: Sameera Perera > Sameera Perera (born 20 August 1988) is a Sri Lankan cricketer. The answer is Sri Lanka.

Q: What screenwriter with credits for "Evolution" co-wrote a film starring Nicolas Cage and Téa Leoni?
A: The Family Man > The Family Man is a 2000 American romantic comedy-drama film starring Nicolas Cage and Téa Leoni. David Weissman > His film credits include "The Family Man" (2000), "Evolution" (2001), and "When in Rome" (2010). The answer is David Weissman.

Q: Ralph Hefferline was a psychology professor at a university that is located in what city?
A: Ralph Hefferline > Ralph Franklin Hefferline was a psychology professor at Columbia University. Columbia University > Columbia University is a private Ivy League research university in Upper Manhattan, New York City. The answer is New York City.

Table 15: Few-shot examples used for CoT-FSP baseline model in Wiki QA domain.



In this task, you are given a middle-school math question (**Question**), along with a gold reference answer (**Gold_Answer**) as its standard solution. There are also 4 candidate answers (**Answer_A/B/C/D**) generated by our AI models A, B, C and D, and your task is to judge whether each candidate answer has errors.

In the annotation, you will do a binary choice selection of **"Yes" or "No"** for each candidate answer for **two** types of errors (definition below). Because there are 4 candidate answers and 2 types of errors. For each question, you will make **a total of 8 binary choices**.

Specifically, we consider **two types of errors**: **1**) **arithmetic error; 2) reasoning error**

1. Whether the answer has any **arithmetic error**? i.e., math derivations are wrongly calculated, or math equations do not hold. Choose Yes or No. For example:

Question: A pirate crew is digging for buried treasure on the island marked X on a map. They dug ten holes the first day, thirteen holes the second day, and eight holes the third day. They stopped digging early on the third day to fill in two holes the pirates kept falling in. How many holes are now on the island?

Candidate answer: On the first day, the pirates dug 10 holes. On the second day, they dug 13 holes. On the third day, they dug 8 holes, and filled in 2 holes, so there are 10 + 13 + 8 - 2 = 27 holes. The answer is 27.

Analysis: Yes, the answer has an arithmetic error, where 10 + 13 + 8 - 2 should be 29.

2. Whether the answer has any **reasoning error**? i.e., misunderstanding of the question, or having a wrong problem solving strategy, which is unrelated to arithmetic correctness. Choose Yes or No. For example:

Question: Marcus is half of Leo's age and five years younger than Deanna. Deanna is 26. How old is Leo?

Candidate answer: Marcus is half of Leo's age. So Marcus is 26 / 2 = 13 years old. Leo is 13 + 5 = 18 years old. The answer is 18.

Analysis: Yes, the answer has a reasoning error, Leo should be (26 - 5) * 2 = 42 years old.

Notes:
1. Please forgive any grammar or spelling typos in all questions and answers, they are not considered as math solution errors.
2. If you feel the gold reference answer (Gold_Answer) is wrong, just ignore it and make the judgment based on your own answer to the question.

Figure 5: Guideline for human evaluation on GSM8K mathematical reasoning.