Exploring Equation as a Better Intermediate Meaning Representation for Numerical Reasoning of Large Language Models

Dingzirui Wang[1], **Longxu Dou**[1], **Wenbin Zhang**[2], **Junyu Zeng**[2], **Wanxiang Che**[1,*]

[1]Harbin Institute of Technology
[2]Yunfu Technology (Beijing) Co., Ltd.
{dzrwang, lxdou, car}@ir.hit.edu.cn, {zhangwenbin, zengjunyu}@yunfutech.com

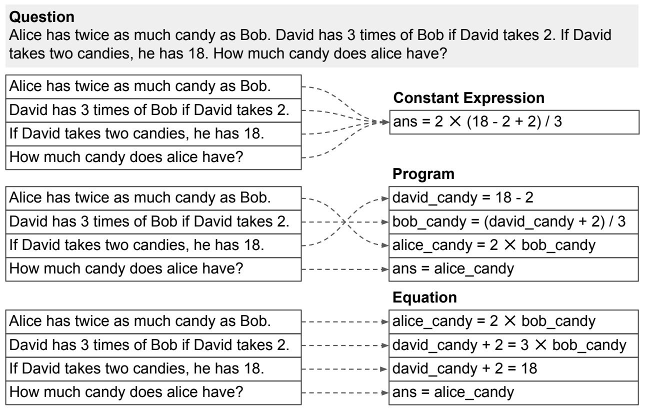
Abstract

Numerical reasoning is a vital capability for natural language processing models to understand and process numerical information in real-world scenarios. Most current methods first generate the Intermediate Meaning Representations (IMRs) of questions and then generate answers. Current SOTA methods generate programs as IMRs with large language models (LLMs). Intuitively, equations have fewer restrictions and closer semantics to the question than programs, leading to higher generation accuracy. However, current LLMs generate equations worse than programs, where we assume that the equation data is rare in pre-training data compared to programs. So in this paper, we try to use equations as IMRs to solve the numerical reasoning task by addressing two problems: *(1)* Theoretically, how to prove that the equation is an IMR with higher generation accuracy than programs; *(2)* Empirically, how to improve the generation accuracy of equations with LLMs. For the first problem, we propose and prove a proposition to theoretically compare the generation accuracy of different IMRs. For the second problem, we present a method called **B**oosting Numerical **R**easoni**n**g by **D**ecomposing the **G**eneration of **E**quations (BRIDGE), which can improve the accuracy of LLMs in generating equations as IMRs by reducing the tendency of generating constant expressions and programs. Our method improves the performance by 2.2%, 0.9%, and 1.7% on GSM8K, SVAMP, and Algebra datasets compared to the previous state-of-the-art methods under the single reasoning path setting. Our code and prompts are available at https://github.com/zirui-HIT/Bridge_for_Numerical_Reasoning.

Introduction

Numerical reasoning is an essential ability of natural language processing (NLP) models to handle documents fulling of numerical information, which is widely used in finance, science, and other fields (Chen et al. 2021b, 2023; Lu et al. 2023). Generally, numerical reasoning is to generate a value result based on the given question, which describes the values and relationships of quantities (Zhang et al. 2020).

Numerical calculations, by their inherent complexity, make it a struggle to produce accurate value results directly (Thawani et al. 2021). To overcome this challenge,


Question
Alice has twice as much candy as Bob. David has 3 times of Bob if David takes 2. If David takes two candies, he has 18. How much candy does alice have?

Alice has twice as much candy as Bob.
David has 3 times of Bob if David takes 2.
If David takes two candies, he has 18.
How much candy does alice have?

Constant Expression
ans = 2 × (18 - 2 + 2) / 3

Alice has twice as much candy as Bob.
David has 3 times of Bob if David takes 2.
If David takes two candies, he has 18.
How much candy does alice have?

Program
david_candy = 18 - 2
bob_candy = (david_candy + 2) / 3
alice_candy = 2 × bob_candy
ans = alice_candy

Alice has twice as much candy as Bob.
David has 3 times of Bob if David takes 2.
If David takes two candies, he has 18.
How much candy does alice have?

Equation
alice_candy = 2 × bob_candy
david_candy + 2 = 3 × bob_candy
david_candy + 2 = 18
ans = alice_candy

Figure 1: Examples of three types of IMRs. The dotted line indicates the correspondence between the question and the generated IMR. The more complex the correspondence is, the more challenging it becomes to generate accurately.

most current methods first generate **I**ntermediate **M**eaning **R**epresentations (**IMRs**) of questions, then compute the value results with external tools (e.g., algorithms, interpreters) (Huang et al. 2018; Wang, Zhang, and Wang 2023). For example, the constant expression is a commonly used IMR (Roy and Roth 2015; Koncel-Kedziorski et al. 2016). The program is another common IMR used by the current state-of-the-art (SOTA) methods (Chen et al. 2022; Gao et al. 2022; Xie et al. 2023). Examples of these two types of IMRs are shown in Figure 1. Current methods mainly use large language models (LLMs) to generate IMRs because LLMs can use few-shot inference to generate various IMRs without training (Jin and Lu 2023; Xie et al. 2023).

In addition to the IMRs above, previous work has also used systems of equations as IMRs (Roy, Upadhyay, and Roth 2016; He-Yueya et al. 2023). From an intuitive view, using equations as IMRs should be better than using programs because equations do not need to define variables before using them, leading to closer semantics to natural language questions. An intuitive example is shown in Figure 1. However, the current method using equations as IMRs does not have better performance than that of using programs (He-Yueya et al. 2023). So in this paper, we try to use equa-

*Corresponding author.

tions as IMRs to solve the numerical reasoning task by addressing two problems: ***(1) Theoretically, how to prove that the equation is an IMR with higher generation accuracy than programs***; ***(2) Empirically, how to improve the generation accuracy of equations with LLMs***.

For the first problem, we present and prove a proposition to compare the generation accuracy of different IMRs: ***given two IMRs, IMR$_A$ and IMR$_B$, if IMR$_A$ is the subset of IMR$_B$, then the accuracy in generating IMR$_B$ of questions theoretically surpasses IMR$_A$.*** Based on this proposition, we can prove that the accuracy of generating equations is higher than that of programs. Because programs can be seen as equations with the restriction "*variables must be defined before being used*", programs are a subset of equations. Consequently, employing equations as IMRs confers theoretically higher generation accuracy than that of programs.

For the second problem, current LLMs have poor performance in generating equations (He-Yueya et al. 2023). We assume that is because current LLMs are mainly pre-trained with constant expressions and programs for numerical reasoning (Brown et al. 2020; Chen et al. 2021a), which makes LLMs prefer to generate these two types of IMRs rather than other IMRs during few-shot inference. This limits the numerical reasoning ability of LLMs since these two types of IMRs may not be the best IMRs for this task. To lower the tendency of LLMs to generate constant expressions and programs, we propose our method called **B**oosting Numerical **R**easoni**n**g by **D**ecomposing the **G**eneration of **E**quations (BRIDGE). Our method erases asking parts and decomposes questions into sub-questions, which can improve the tendency of LLMs to generate equations.

To evaluate the effectiveness of BRIDGE, we adopt experiments on GSM8K (Cobbe et al. 2021), SVAMP (Patel, Bhattamishra, and Goyal 2021), and Algebra (He-Yueya et al. 2023), which are mainstream datasets of the numerical reasoning task. BRIDGE improves 1.6% performance over the previous SOTA results on all above datasets on average and achieves new SOTA results under the single reasoning path setting. In addition, ablation experiments show that BRIDGE can improve the proportion of equations in generated results, which shows that our method can indeed improve the tendency of LLMs to generate equations as IMRs.

Our contribution can be summarized as follows:

- To theoretically prove that equations have higher generation accuracy than the IMRs of the current SOTA methods, we present and prove a proposition that can theoretically compare the generation accuracy of different IMRs.
- To empirically improve the performance of LLMs in generating equations other than constant expressions and programs, we present BRIDGE, which improves the tendency of LLMs to generate equations as IMRs.
- To verify the effectiveness of BRIDGE, we conduct experiments on multiple mainstream numerical reasoning datasets, where our method achieves new SOTA results on all datasets under the single reasoning path setting.

Methodology

In this section, we introduce our work in detail. First, we explain why we use equations as IMRs as an intuitive explanation for our proposition. Then, we present and prove a proposition that can theoretically compare the generation accuracy of different IMRs. After that, we introduce the pipeline of BRIDGE.

Equations as Intermediate Meaning Representation

Intuitive Principle of IMR Design The previous research has shown that even with the same model architecture, generating different IMRs may lead to different performances (Huang et al. 2018; Li et al. 2022). Therefore, the design of IMRs also affects the accuracy of generating. Generally, the more restrictive rules there are on IMRs, the harder it is for the model to generate such IMRs. Because if IMRs have more restrictions, the model needs more reasoning steps to meet these restrictions, resulting in the semantic difference with the question, increasing the difficulty of reasoning.

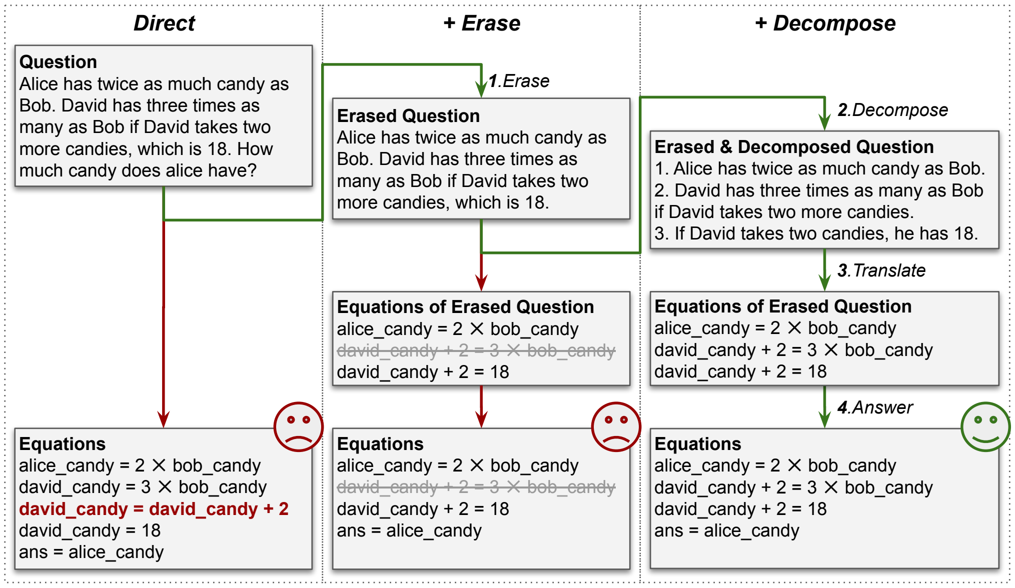
Comparison of Different IMRs A commonly used IMR is the constant expression (Roy and Roth 2015; Koncel-Kedziorski et al. 2016), which asks that only values, no variables, appear in one single expression as a result. This leads to a great semantic difference between the questions and the constant expressions because questions may describe many relationships between different quantities. To address these restrictions, previous works propose using programs, which is the IMR of the current SOTA methods (Chen et al. 2022; Gao et al. 2022; Xie et al. 2023). First, the program allows to use variables to calculate other variables instead of just constant values. Second, programs can use multiple statements rather than just one statement to represent the answer.

However, programs also have their restrictions, where each variable must be defined before use. This also leads to a semantic difference between the IMR and the question because one question may describe the relationships of variables before giving their values. To solve this restriction, we propose to use the equation as IMR, where the equation allows variables to be used before their definition (Roy, Upadhyay, and Roth 2016; He-Yueya et al. 2023). Therefore, the equation is closer to the semantics of questions than programs, leading to higher generation accuracy.

Discovery from IMR Comparison From the above analysis, we can discover that increasing the number of restrictions on IMRs is actually a process of screening a set. For example, programs are equations with the restriction "*variables must be defined before being used*". Constant expressions are programs with the restriction "*there is only one assignment statement with only specific values*". In the following, we summarize this discovery into a proposition to guide the design of IMRs with high generation accuracy.

Generation Accuracy of Different Intermediate Meaning Representations

In the following, we prove that generating equations has higher accuracy than generating programs, for which we



Direct
+ Erase
+ Decompose
Question
Alice has twice as much candy as Bob. David has three times as many as Bob if David takes two more candies, which is 18. How much candy does alice have?
1.Erase
Erased Question
Alice has twice as much candy as Bob. David has three times as many as Bob if David takes two more candies, which is 18.
2.Decompose
Erased & Decomposed Question
1. Alice has twice as much candy as Bob.
2. David has three times as many as Bob if David takes two more candies.
3. If David takes two candies, he has 18.
Equations of Erased Question
alice_candy = 2 × bob_candy
~~david_candy + 2 = 3 × bob_candy~~
david_candy + 2 = 18
3.Translate
Equations of Erased Question
alice_candy = 2 × bob_candy
david_candy + 2 = 3 × bob_candy
david_candy + 2 = 18
4.Answer
Equations
alice_candy = 2 × bob_candy
david_candy = 3 × bob_candy
david_candy = david_candy + 2
david_candy = 18
ans = alice_candy
Equations
alice_candy = 2 × bob_candy
~~david_candy + 2 = 3 × bob_candy~~
david_candy + 2 = 18
ans = alice_candy
Equations
alice_candy = 2 × bob_candy
david_candy + 2 = 3 × bob_candy
david_candy + 2 = 18
ans = alice_candy

Figure 2: The illustration of BRIDGE under different settings of reasoning stages. The incorrect reasoning paths and results are annotated with red. The Direct method modifies the value of the constant unknown "$david\_candy$" like the program. The method that only uses Erase misses the equation "$david\_candy + 2 = 3 \times bob\_candy$" since it pretends to translate one entire sentence into one single equation. The correct one is annotated with green, which decomposes the numerical reasoning of LLMs into four stages. (1) Erase: erase the asking part of the question; (2) Decompose: decompose the question into multiple sub-questions; (3) Translate: translate the sub-questions to equations; (4) Answer: generate the answer equation.

present a proposition that can theoretically compare the generation accuracy of different IMRs. Before presenting this proposition, we first propose an auxiliary proposition:

Proposition 1 *Given IMR$_A$ and IMR$_B$, let A = {all examples of IMR_A} and B = {all examples of IMR_B}, $A \subseteq B$. Given one natural language question q. Let $N(q, x)$ denote the hop number (Yang et al. 2018) required to generate x based on q. Then $\exists b \in B, \forall a \in A, N(q, b) \leq N(q, a)$, where all a, b are the IMRs of q.*

The hop number in Proposition 1 can be regarded as a numerical quantification of the difficulty of generating different IMRs. Intuitively, if IMR$_A$ is a subset of IMR$_B$, it means that IMR$_A$ has more restrictions than IMR$_B$, so more reasoning hops are needed to convert a natural language question to the corresponding IMR$_A$. With Proposition 1, we can present the proposition to compare the generation accuracy of different IMRs:

Proposition 2 *Given IMR$_A$ and IMR$_B$, $A \subseteq B$. Given one natural language question q. Then generating IMR$_B$ of q has higher accuracy than IMR$_A$.*

Based on Proposition 2, we can compare the generation accuracy of different IMRs by judging the inclusion relationship of IMRs. Considering the discussion above, programs are a subset of equations, so generating equations as IMRs has higher accuracy than generating programs in theory. However, Proposition 2 can only compare the generation accuracy from the view of different IMRs themselves. Apart from the type of IMR, the generation accuracy also depends on many other factors, such as the model architecture and pre-training data. For example, although using equations as IMR has higher accuracy than constant expressions according to Proposition 2, models pre-trained with constant expressions have higher accuracy in generating expressions than generating equations. Therefore, in order to empirically enhance the ability of LLMs to generate equations, we need to design specific methods that boost the generation capabilities of LLMs for equations outside of constant expressions and programs.

Pipeline of BRIDGE

In this section, we introduce the pipeline of BRIDGE, which decomposes the numerical reasoning into four stages. The illustration of BRIDGE is shown in Figure 2.

Stage1: Erase The previous research has shown that the current NLP models mainly learn the mapping between input and output formats rather than specific NLP capabilities (McCoy, Pavlick, and Linzen 2019; Jawahar, Sagot, and Seddah 2019; Bubeck et al. 2023). So during the few-shot inference, current LLMs are more inclined to generate constant expressions and programs since we assume that these IMRs are mainly contained in the pre-training data (Brown et al. 2020; Chen et al. 2021a). However, since the pre-training data is not available as an open resource, we have been unable to validate this assumption thus far. To enhance the tendency of current LLMs to generate equations, we should disrupt the input format of numerical reasoning questions that LLMs have seen in the pre-training data

We observe that even if the asking part is erased, the remaining part can still be expressed as solvable equations.

Algorithm 1: Generate Answer Equation

```
Input: Natural language question $question$, translated equations $equations$, number of retries limit $retry$.
Output: Equation of the answer.
1:  $temperature = 0$
2:  for $i = 1$ to $retry$ do
3:      $ans$ = answer($question$, $equations$, $temperature$)
4:      if is_solvable($equations$, $ans$) then
5:          return $ans$
6:      else
7:          $temperature$ += $0.1$
8:      end if
9:  end for
10: return None
```

For example, about the question "*Alice has twice as much candy as Bob. How much candy does Alice have if Bob has 12 candies*", after erasing the asking part "*How much candy does Alice have*", the rest part can still be listed as equations "$alice\_candy = 2\times bob\_candy, bob\_candy = 12$". To guide the LLMs in generating equations, we erase the asking part of each question. This disrupts the input format, as observed in the pre-training data, and reduces the tendency to generate constant expressions or programs.

Stage2: Decompose During generating equations based on questions, LLMs may translate one entire sentence into one single equation, resulting in missing intermediate information and unsolvable equations. Take the results only with Erase stage as an example in Figure 2, LLMs directly translate "*David has three times as many as Bob if David takes two more candies, which is 18*" into "$david\_candy + 2 = 18$" while ignoring the information "*David has three times as many as Bob if David takes two more candies*". To address this issue, we try to decompose the question into sub-equations before generating the equations. With the decomposed sub-questions, LLMs are able to generate equations based on finer-grained information, thus alleviating the phenomenon of missing information.

Stage3: Translate In this stage, we use both the erased question and the decomposed sub-questions as the input to generate the corresponding equations with LLMs, which complement each other for the generation of the complete equations. However, the Translation stage may generate unsolvable equations where there is no solution for the equations. This makes it to be unable to get the value of each quantity to calculate the answer in the next stage. To address this problem, if the Translation stage generates unsolvable equations, we set the result to be empty and let the next stage generate all equations from scratch.

Stage4: Answer The Translation stage currently computes the value of each quantity separately, whereas we need to collectively compute them together, which is done by equation solving and post-processing [1]. Concretely, in this stage, we use the original questions and the translated equations as input and output the final answer equation. The process of this stage is shown in Algorithm 1. Since the translated equations themselves have certain semantic information, LLMs can understand the meaning of each unknown and use them to represent the answer equation. If the equations are unsolvable, we can not get the answer. So we let the LLMs regenerate the results if the generated equations are unsolvable and gradually increase the generation temperature until the results are solvable.

Experiments

In this section, we adopt experiments to verify the effectiveness of our method. First, we give the setup information of our experiments. Then, we present the main experiment results to demonstrate that our method can improve the numerical reasoning ability of LLMs. After that, to prove the effectiveness of each stage of BRIDGE, we give the ablation experiments of each stage. Finally, we analyze the experiment results of BRIDGE to shed light on future research.

Experiment Setup

Dataset We adopt BRIDGE to GSM8K (Cobbe et al. 2021), SVAMP (Patel, Bhattamishra, and Goyal 2021), and Algebra (He-Yueya et al. 2023), which are widely used numerical reasoning datasets. The question number of GSM8K, SVAMP, and Algebra is 1319, 1000, and 222, respectively. GSM8K consists of grade school math questions, which require 2-8 steps of reasoning and use basic arithmetic operations ($+-\times\div$) to get answers. SVAMP contains $1,000$ math questions selected from the existing numerical reasoning datasets, using basic arithmetic operations to solve. Different from the above datasets, Algebra is a dataset containing more algebra questions, which can reflect the ability of the model to translate the questions into the corresponding algebraic equations.

Metric We use the exact match (EM) as the evaluation metric of our work. Because of the round-off error, following the previous work (Chen et al. 2022; Gao et al. 2022), we consider the prediction is equal to the ground truth if their relative difference is below 10^{-3}.

Model We use Codex (Chen et al. 2021a) and GPT3.5 [2] as our experimental LLMs, which belong to the most widely used LLMs. Codex is an advanced model that is capable of translating natural language instructions into code across a variety of programming languages. GPT3.5 is the model improved on GPT-3 (Brown et al. 2020) and can handle both natural language and code.

Implement Detail For the equation generation, we use the Azure OpenAI API of `code-davinci-002` and `gpt-3.5-turbo` for our experiments [3]. We use Codex to denote `code-davinci-002` and GPT3.5 to denote `gpt-3.5-turbo` for brief in the following. We use 5-8

[1]For example, if the answer should be present with a percentage, the result value needs to be multiplied by 100.

[2]https://platform.openai.com/docs/models/gpt-3-5

[3]https://azure.microsoft.com/en-us/products/cognitive-services/openai-service

shots for all the experimental datasets with different hardness. For equation solving, we employ sympy (Meurer et al. 2017) to solve generated equations, which is a Python package for symbolic mathematics.

Main Result

The main experiment results are shown in Table 1. We can see that BRIDGE brings robust performance improvement, leading to new SOTA results on all datasets and models under the single reasoning path setting, which proves the effectiveness of our method.

GSM8K Compared with the results of the previous SOTA methods, BRIDGE consistently surpasses the previous methods using programs as IMRs (PoT, PAL), empirically proving that equations are a better IMR than programs and the correctness of Proposition 2 to a certain extent.

SVAMP BRIDGE also achieves the new SOTA result with 0.9% improvement. Notably, we can observe that the performance boost brought by BRIDGE in SVAMP is not as valid as GSM8K. This is because the semantics of bad cases in previous methods are complicated, requiring enhancing the understanding ability of LLMs to relationships of quantities rather than simply changing IMRs.

Algebra BRIDGE brings 1.7% improvement compared with the previous SOTA result. However, our method is less robust compared with other datasets, with a fluctuation of more than 1.7%. It is because Algebra is smaller than the other two datasets with only 222 questions, resulting in more obvious performance fluctuations.

Our method can also apply the self-consistency method (Wang et al. 2023) to enhance the performance. To prove this point, we evaluate BRIDGE on GSM8K with self-consistency, which is shown in Table 3. The result shows that self-consistency also brings significant improvement to our method.

Ablation Study

Answer Generation We adopt the ablation study to verify the stage effectiveness of BRIDGE. The experiment result is shown in Table 4, from which we can see that: *(1)* The Erase and Decompose stages can bring improvement to all datasets, proving the effectiveness of these two stages; *(2)* Compared with Erase, using Decompose can bring a more significant improvement on GSM8K and SVAMP since Decompose is also effective for questions that do not require equation solving, while Erase is mainly designed for questions using equations; *(3)* Compared with GSM8K, the improvement in SVAMP is less obvious because the questions of SVAMP are much simpler than GSM8K, and most questions not solved by previous methods require enhancing abilities other than numerical reasoning, so the improvement brought by our method is not significant.

Equation Generation To verify whether the Erase stage can improve the tendency of LLMs to generate equations, we count the number of generated equations in the results with and without Erase. The experiment results are shown in Table 2. From the results, we can see that: *(1)* On each dataset, the numbers of results using equations on both correct and total cases have increased, proving that the Erase stage indeed improves the tendency of LLMs to generate equations; *(2)* Compared with GSM8K, the improvement of the Erase stage in Algebra is not obvious because Algebra mainly includes algebra questions, which can guide to generate algebra equations by questions themselves.

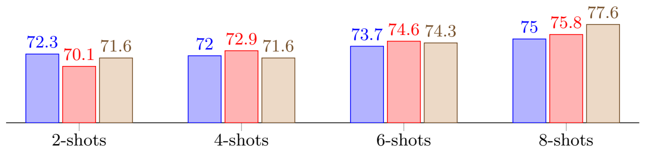


Figure 3: The exact match of inference with different shots on GSM8K run three times using `gpt-3.5-turbo`. Different color denotes the result of the different run.

Analysis

Sensitivity to Exemplars To study the impact of different prompts on BRIDGE, we write a total of 15 samples and randomly select {2, 4, 6, 8} shots from them as prompts to run three times. The experiment results are shown in Figure 3. From the results, we can see that as the number of shots in the prompt increases, the performance of our method gradually improves. Besides, as the number of shots increases, the variance of the results is gradually decreasing, indicating that the robustness is also gradually improving.

Equation Proportion To evaluate whether BRIDGE enhances the numerical reasoning ability of LLMs by generating equations, we select the cases where PoT is incorrect while our method is correct and see if each case generates equations as IMRs. The proportions of generating equations of different methods and datasets are shown in Table 5. Based on the results, we can see that: *(1)* On all models and all datasets, the performance improvement has equations involved, and it is most significant in Algebra, where about half of the improvement uses equations; *(2)* The performance improvement is not all brought by using equations as IMRs because the results of LLMs are not robust, and many questions in the part that does not use equations may be solved in the results of other runs of PoT; *(3)* The improvement brought by using equations on SVAMP is not significant because the questions of this dataset are relatively simple, and most of them can be solved without equations.

Reasoning Complex We analyze the effectiveness of our method of solving the questions with different complexes, which are categorized by the number of statements or equations of answers. The performance improvement of our method compared with PoT is shown in Table 6, which shows that: *(1)* Our method can bring performance improvement of questions under different complexes, especially on the Algebra dataset, improvement is more than 30%, which proves the effectiveness of our method under different complexes; *(2)* On the GSM8K dataset, the improvement of more complex questions is not obvious, show-

Model	Method	GSM8K	SVAMP	Algebra
Codex	CoT (Wei et al. 2022)	65.6	74.8	47.9
	Tab-CoT (Jin and Lu 2023)	61.6	82.9	–
	Declarative (He-Yueya et al. 2023)	69.4	–	76.3
	PoT (Chen et al. 2022)	71.6	85.2	–
	PAL (Gao et al. 2022)	72.0	79.4	56.2
	BRIDGE	$\mathbf{74.2 \pm 0.4}$	$\mathbf{86.1 \pm 0.5}$	$\mathbf{78.5 \pm 1.7}$
GPT3.5	CoT (Our runs)	76.5	80.8	53.6
	PoT (Our runs)	74.8	79.3	64.0
	BRIDGE	$\mathbf{77.2 \pm 0.4}$	$\mathbf{82.3 \pm 0.6}$	$\mathbf{82.0 \pm 0.9}$

Table 1: The exact match under different datasets and models with different prompt methods. The results of our method are averaged over five runs. The best results of different datasets are annotated in bold.

Model	Method	GSM8K		SVAMP		Algebra	
		C	T	C	T	C	T
Codex	BRIDGE	83	108	49	57	68	80
	- Erase	61	89	44	52	65	77
	$\Delta(\%)$	36.1	21.3	11.4	9.6	4.6	3.9
GPT3.5	BRIDGE	93	139	56	65	77	86
	- Erase	65	96	43	51	75	81
	$\Delta(\%)$	43.1	44.8	30.2	27.5	2.7	6.2

Table 2: The equations generated with and without Erase stage. C denotes the correct cases, and T denotes the total cases. Δ denotes the ratio of the equations increased after using Erase to that without Erase.

Method	GSM8K
CoT w. Self-Consistency	82.0
PoT w. Self-Consistency	77.4
BRIDGE	77.2
w. Self-Consistency	**83.6** (+6.4)

Table 3: The exact match of `gpt-3.5-turbo` with different prompt methods. The best result is annotated with bold.

ing that BRIDGE brings relatively small performance improvement when dealing with more complex questions.

Error Type In order to investigate the main sources of error in BRIDGE, we count the number of bad cases where errors occurred in different stages. We randomly select one hundred bad cases of GSM8K and SVAMP, then manually classify them according to the stages where the error occurred. The results are shown in Figure 4.

Based on the results, we can draw the conclusion that: *(1)* There are very few cases of LLMs making mistakes in the Erase and Decompose stages because LLMs in these two stages mainly rely on understanding the meaning of the question and do not need reasoning, so they have better performance; *(2)* The errors mainly concentrate in the Translate and Answer stages, which proves that the ability of current LLMs to translate natural language into equations is still weak because LLMs still confuse the semantics between questions and equations after erasing and decomposing; *(3)* The main error types of GSM8K and SVAMP are not consistent because the relationship of quantities in the questions of GSM8K is more complicated and hard to be translated into equations, while the main difficulty of SVAMP lies in determining which quantity is asked by the question.

Model	Method	GSM8K	SVAMP	Algebra
Codex	BRIDGE	74.2	86.1	78.5
	- Erase	69.0	86.0	72.1
	- Decompose	68.2	85.6	77.8
GPT3.5	BRIDGE	77.0	82.5	82.9
	- Erase	76.1	81.9	81.5
	- Decompose	74.2	81.9	79.7

Table 4: The ablation experiment results in Erase and Decompose stages on GSM8K and SVAMP of different models. The first line of each model denotes using BRIDGE.

Model	GSM8K	SVAMP	Algebra
Codex	20.8%	15.3%	47.8%
GPT3.5	41.7%	10.0%	59.6%

Table 5: Among the cases where PoT is incorrect while BRIDGE is correct, the proportion of cases using equations.

Case Study To better understand how BRIDGE improves the numerical reasoning performance of LLMs, we present a case study in Figure 5. Since the program needs to define variables before use, when dealing with the question in Figure 5, it is necessary first to understand the relationships of different quantities in the question, then adjust the order of each sub-question in the result, which is with low generation accuracy. While BRIDGE uses equations as IMRs, the generated results correspond to each sentence of the original question in sequence, and the difficulty of generation is low, leading to the correct result.

#Steps	GSM8K			Algebra		
	PoT	BRIDGE	Δ	PoT	BRIDGE	Δ
≤ 4	71.1	78.6	7.5	49.5	80.5	31.0
$[5, 6]$	71.6	78.3	6.7	20.0	60.0	40.0
≥ 7	68.5	70.0	1.5	5.9	58.8	52.9

Table 6: The exact match of questions with different equations steps of `code-davinci-002`. #Steps denotes the number of answer equations.

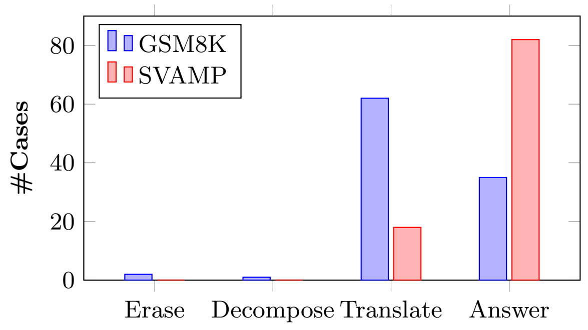


Figure 4: The number of bad cases in different stages of GSM8K and SVAMP of `code-davinci-002`.

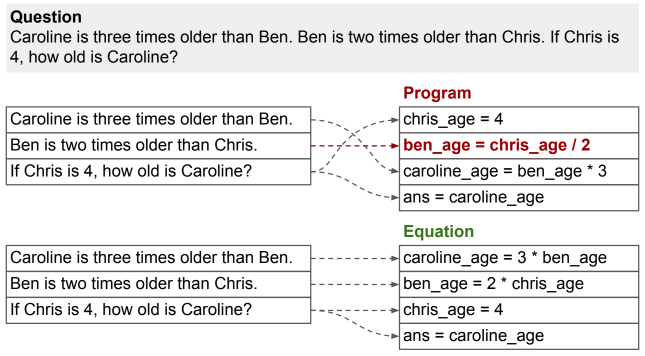

Question
Caroline is three times older than Ben. Ben is two times older than Chris. If Chris is 4, how old is Caroline?
Program
Caroline is three times older than Ben.
Ben is two times older than Chris.
If Chris is 4, how old is Caroline?
chris_age = 4
ben_age = chris_age / 2
caroline_age = ben_age * 3
ans = caroline_age
Equation
Caroline is three times older than Ben.
Ben is two times older than Chris.
If Chris is 4, how old is Caroline?
caroline_age = 3 * ben_age
ben_age = 2 * chris_age
chris_age = 4
ans = caroline_age

Figure 5: The results of PoT and BRIDGE of a case in GSM8K. The above part is the result of PoT, and the below part is BRIDGE. The error steps and results are annotated in red, and the correct results are annotated in green.

Related Work

Numerical Reasoning with LLMs

The ability to perform numerical reasoning is essential for NLP models as it enables them to understand and manipulate numerical information embedded in natural language (Lu et al. 2023). Utilizing LLMs for numerical reasoning tasks has become the mainstream in current research due to their brilliant few-shot inference ability without training (Wei et al. 2022; Chen et al. 2022; Xie et al. 2023).

Early researches guide LLMs to generate answers and the reasoning process in one step (Jie and Lu 2023; Liu and Low 2023; Imani, Du, and Shrivastava 2023). For example, CoT (Wei et al. 2022) asks LLMs to generate the reasoning process with natural language to get the answer. As the pre-training data of LLMs include numerous programs, PoT (Chen et al. 2022) and PAL (Gao et al. 2022) ask the model to generate programs to solve numerical questions. Following the same reason, Tab-CoT (Jin and Lu 2023) asks the model to generate tables to solve this task, as a large amount of tabular data is used during the pre-training of LLMs.

Besides, many works also try to decompose the process of numerical reasoning into multiple steps to reduce the difficulty of model inference (Gaur and Saunshi 2023; Li et al. 2023; Wang, Zhang, and Wang 2023). For example, PHP (Zheng et al. 2023) generates answers through multi-turn inferences, and each turn can use the answers of the previous turns as a hint. Decomposition (Xie et al. 2023) generates multiple candidate results based on the previous steps of each reasoning step (e.g., one program statement) and keeps the best k of them like beam-search.

However, due to the high proportion of constant expressions and programs in the pre-training data, LLMs prefer to generate these two types of IMRs during few-shot inference. These may limit the numerical reasoning capability of LLMs since these IMRs may not be the best format for solving the numerical reasoning task. To overcome this problem, we present BRIDGE, which erases asking parts in questions. After erasing, we disrupt the input structure that LLMs have seen in the pre-training data, and LLMs are less likely to generate constant expressions or programs.

Reasoning with Intermediate Meaning Representation

In the field of NLP, a common way to reduce the difficulty of reasoning is to generate answers with an intermediate meaning representation (IMR) (Gan et al. 2021; Nie et al. 2022; Paul et al. 2023). Such methods first generate IMRs of questions and then use external tools (e.g., algorithms, interpreters) to generate the answer results based on IMRs. Since generating the answer from IMRs is deterministic, the main bottleneck is how to generate IMRs of questions.

The most widely studied IMRs are the semantic representation language (Kamath and Das 2019). Previous works have designed many semantic representation languages to represent natural language sentences, such as AST (Jones 2003) and AMR (Banarescu et al. 2013). By converting into these languages, then applying the converted results on downstream tasks, the semantic understanding ability of the model can be effectively improved (Che et al. 2021).

Many methods also employ IMRs to solve the numerical reasoning task (Wang, Zhang, and Wang 2023; Paul et al. 2023). A commonly used IMR is constant expressions (Roy and Roth 2015; Koncel-Kedziorski et al. 2016). The current SOTA methods use programs as IMRs because programs have closer semantics to questions than constant expressions, and current LLMs have strong program generation capabilities (Chen et al. 2021a, 2022; Gao et al. 2022). Besides, there are many other types of IMRs, such as dolphin languages (Huang et al. 2018), domain-specific languages (Chen et al. 2021b) and equations (Roy, Upadhyay, and Roth 2016; He-Yueya et al. 2023).

However, most methods design IMRs based on expert experience or experimental results, affected by factors other than IMRs themselves, such as model structure and training data. To theoretically compare the generation of different IMRs, in this paper, we propose and prove a proposition to guide the design of IMRs with high generation accuracy.

Conclusion

In this paper, we employ equations as IMRs to solve the numerical reasoning task by addressing two problems: *(1)* Theoretically, how to prove that the equation is an IMR with higher generation accuracy than programs; *(2)* Empirically, how to improve the generation accuracy of equations with LLMs. For the first problem, we present and prove a proposition to compare the generation accuracy of different IMRs in theory. For the second problem, we present BRIDGE to enhance the equation generation of LLMs by reducing the tendency of generating constant expressions and programs and decomposing questions. To evaluate BRIDGE, we conduct experiments across three datasets: GSM8K, SVAMP, and Algebra. Compared to the previous SOTA results, our method has increased average performance by 1.6%, setting new SOTA performance across all the datasets under the single reasoning path setting. Moreover, ablation experiments show that using BRIDGE can enhance the tendency of LLMs to generate equations as IMRs, proving that our method can improve the ability of LLMs to generate IMRs outside of constant expressions and programs.

Acknowledgments

We thank all anonymous reviewers for their constructive comments. We gratefully acknowledge the support of the National Natural Science Foundation of China (NSFC) via grant 62236004 and 62206078, and the support of Du Xiaoman (Beijing) Science Technology Co., Ltd.

References

Banarescu, L.; Bonial, C.; Cai, S.; Georgescu, M.; Griffitt, K.; Hermjakob, U.; Knight, K.; Koehn, P.; Palmer, M.; and Schneider, N. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 178–186. Sofia, Bulgaria: Association for Computational Linguistics.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 1877–1901. Curran Associates, Inc.

Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y. T.; Li, Y.; Lundberg, S.; Nori, H.; Palangi, H.; Ribeiro, M. T.; and Zhang, Y. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. arXiv:2303.12712.

Che, W.; Feng, Y.; Qin, L.; and Liu, T. 2021. N-LTP: An Open-source Neural Language Technology Platform for Chinese. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 42–49. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; de Oliveira Pinto, H. P.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; Ray, A.; Puri, R.; Krueger, G.; Petrov, M.; Khlaaf, H.; Sastry, G.; Mishkin, P.; Chan, B.; Gray, S.; Ryder, N.; Pavlov, M.; Power, A.; Kaiser, L.; Bavarian, M.; Winter, C.; Tillet, P.; Such, F. P.; Cummings, D.; Plappert, M.; Chantzis, F.; Barnes, E.; Herbert-Voss, A.; Guss, W. H.; Nichol, A.; Paino, A.; Tezak, N.; Tang, J.; Babuschkin, I.; Balaji, S.; Jain, S.; Saunders, W.; Hesse, C.; Carr, A. N.; Leike, J.; Achiam, J.; Misra, V.; Morikawa, E.; Radford, A.; Knight, M.; Brundage, M.; Murati, M.; Mayer, K.; Welinder, P.; McGrew, B.; Amodei, D.; McCandlish, S.; Sutskever, I.; and Zaremba, W. 2021a. Evaluating Large Language Models Trained on Code. arXiv:2107.03374.

Chen, W.; Ma, X.; Wang, X.; and Cohen, W. W. 2022. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. arXiv:2211.12588.

Chen, W.; Yin, M.; Ku, M.; Lu, P.; Wan, Y.; Ma, X.; Xu, J.; Wang, X.; and Xia, T. 2023. TheoremQA: A Theorem-driven Question Answering dataset. arXiv:2305.12524.

Chen, Z.; Chen, W.; Smiley, C.; Shah, S.; Borova, I.; Langdon, D.; Moussa, R.; Beane, M.; Huang, T.-H.; Routledge, B.; and Wang, W. Y. 2021b. FinQA: A Dataset of Numerical Reasoning over Financial Data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3697–3711. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168.

Gan, Y.; Chen, X.; Xie, J.; Purver, M.; Woodward, J. R.; Drake, J.; and Zhang, Q. 2021. Natural SQL: Making SQL Easier to Infer from Natural Language Specifications. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2030–2042. Punta Cana, Dominican Republic: Association for Computational Linguistics.

Gao, L.; Madaan, A.; Zhou, S.; Alon, U.; Liu, P.; Yang, Y.; Callan, J.; and Neubig, G. 2022. PAL: Program-aided Language Models. *arXiv preprint arXiv:2211.10435*.

Gaur, V.; and Saunshi, N. 2023. Reasoning in Large Language Models Through Symbolic Math Word Problems. In *Findings of the Association for Computational Linguistics: ACL 2023*, 5889–5903. Toronto, Canada: Association for Computational Linguistics.

He-Yueya, J.; Poesia, G.; Wang, R. E.; and Goodman, N. D.

2023. Solving Math Word Problems by Combining Language Models With Symbolic Solvers. arXiv:2304.09102.

Huang, D.; Yao, J.-G.; Lin, C.-Y.; Zhou, Q.; and Yin, J. 2018. Using Intermediate Representations to Solve Math Word Problems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 419–428. Melbourne, Australia: Association for Computational Linguistics.

Imani, S.; Du, L.; and Shrivastava, H. 2023. MathPrompter: Mathematical Reasoning using Large Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, 37–42. Toronto, Canada: Association for Computational Linguistics.

Jawahar, G.; Sagot, B.; and Seddah, D. 2019. What Does BERT Learn about the Structure of Language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3651–3657. Florence, Italy: Association for Computational Linguistics.

Jie, Z.; and Lu, W. 2023. Leveraging Training Data in Few-Shot Prompting for Numerical Reasoning. In *Findings of the Association for Computational Linguistics: ACL 2023*, 10518–10526. Toronto, Canada: Association for Computational Linguistics.

Jin, Z.; and Lu, W. 2023. Tab-CoT: Zero-shot Tabular Chain of Thought. arXiv:2305.17812.

Jones, J. 2003. Abstract Syntax Tree Implementation Idioms. *Pattern Languages of Program Design*. Proceedings of the 10th Conference on Pattern Languages of Programs (PLoP2003) http://hillside.net/plop/plop2003/papers.html.

Kamath, A.; and Das, R. 2019. A Survey on Semantic Parsing. In *Automated Knowledge Base Construction (AKBC)*.

Koncel-Kedziorski, R.; Roy, S.; Amini, A.; Kushman, N.; and Hajishirzi, H. 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*, 1152–1157. San Diego, California: Association for Computational Linguistics.

Li, Y.; Lin, Z.; Zhang, S.; Fu, Q.; Chen, B.; Lou, J.-G.; and Chen, W. 2023. Making Language Models Better Reasoners with Step-Aware Verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 5315–5333. Toronto, Canada: Association for Computational Linguistics.

Li, Z.; Guo, J.; Liu, Q.; Lou, J.-G.; and Xie, T. 2022. Exploring the Secrets Behind the Learning Difficulty of Meaning Representations for Semantic Parsing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 3616–3625. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics.

Liu, T.; and Low, B. K. H. 2023. Goat: Fine-tuned LLaMA Outperforms GPT-4 on Arithmetic Tasks. arXiv:2305.14201.

Lu, P.; Qiu, L.; Yu, W.; Welleck, S.; and Chang, K.-W. 2023. A Survey of Deep Learning for Mathematical Reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 14605–14631. Toronto, Canada: Association for Computational Linguistics.

McCoy, T.; Pavlick, E.; and Linzen, T. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3428–3448. Florence, Italy: Association for Computational Linguistics.

Meurer, A.; Smith, C. P.; Paprocki, M.; Čertík, O.; Kirpichev, S. B.; Rocklin, M.; Kumar, A.; Ivanov, S.; Moore, J. K.; Singh, S.; Rathnayake, T.; Vig, S.; Granger, B. E.; Muller, R. P.; Bonazzi, F.; Gupta, H.; Vats, S.; Johansson, F.; Pedregosa, F.; Curry, M. J.; Terrel, A. R.; Roučka, v.; Saboo, A.; Fernando, I.; Kulal, S.; Cimrman, R.; and Scopatz, A. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3: e103.

Nie, L.; Cao, S.; Shi, J.; Sun, J.; Tian, Q.; Hou, L.; Li, J.; and Zhai, J. 2022. GraphQ IR: Unifying the Semantic Parsing of Graph Query Languages with One Intermediate Representation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 5848–5865. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics.

Patel, A.; Bhattamishra, S.; and Goyal, N. 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*, 2080–2094. Online: Association for Computational Linguistics.

Paul, D.; Ismayilzada, M.; Peyrard, M.; Borges, B.; Bosselut, A.; West, R.; and Faltings, B. 2023. REFINER: Reasoning Feedback on Intermediate Representations. arXiv:2304.01904.

Roy, S.; and Roth, D. 2015. Solving General Arithmetic Word Problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1743–1752. Lisbon, Portugal: Association for Computational Linguistics.

Roy, S.; Upadhyay, S.; and Roth, D. 2016. Equation Parsing : Mapping Sentences to Grounded Equations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1088–1097. Austin, Texas: Association for Computational Linguistics.

Thawani, A.; Pujara, J.; Ilievski, F.; and Szekely, P. 2021. Representing Numbers in NLP: a Survey and a Vision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 644–656. Online: Association for Computational Linguistics.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q. V.; Chi, E. H.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.

Wang, Y.; Zhang, Z.; and Wang, R. 2023. Meta-Reasoning: Semantics-Symbol Deconstruction For Large Language Models. arXiv:2306.17820.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; brian ichter; Xia, F.; Chi, E. H.; Le, Q. V.; and Zhou, D. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Xie, Y.; Kawaguchi, K.; Zhao, Y.; Zhao, X.; Kan, M.-Y.; He, J.; and Xie, Q. 2023. Decomposition Enhances Reasoning via Self-Evaluation Guided Decoding. arXiv:2305.00633.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2369–2380. Brussels, Belgium: Association for Computational Linguistics.

Zhang, J.; Wang, L.; Lee, R. K.-W.; Bin, Y.; Wang, Y.; Shao, J.; and Lim, E.-P. 2020. Graph-to-Tree Learning for Solving Math Word Problems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 3928–3937. Online: Association for Computational Linguistics.

Zheng, C.; Liu, Z.; Xie, E.; Li, Z.; and Li, Y. 2023. Progressive-Hint Prompting Improves Reasoning in Large Language Models. arXiv:2304.09797.