Forward-Backward Reasoning in Large Language Models for Mathematical Verification

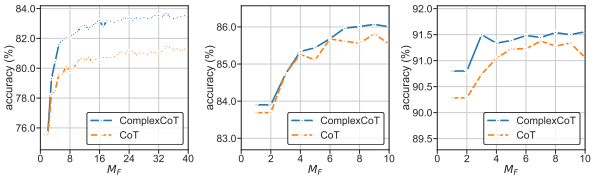
Anonymous ACL submission

Abstract

Self-Consistency samples diverse reasoning chains with answers and chooses the final answer by majority voting. It is based on forward reasoning and cannot further improve performance by sampling more reasoning chains when saturated. To further boost performance, we introduce backward reasoning to verify candidate answers. Specifically, for mathematical tasks, we mask a number in the question and ask the LLM to answer a backward question created by a simple template, i.e., to predict the masked number when a candidate answer is provided. Instead of using forward or backward reasoning alone, we propose FOBAR to combine FOrward and BAckward Reasoning for verification. Extensive experiments on six standard mathematical data sets and three LLMs show that FOBAR achieves state-of-the-art performance. In particular, FOBAR outperforms Self-Consistency, which uses forward reasoning alone, demonstrating that combining forward and forward reasoning is better. In addition, FOBAR performs better than existing verification methods, showing the effectiveness of the simple template used in backward reasoning and the proposed combination. Extensions to non-mathematical problems are also discussed and validated empirically.

1 Introduction

Pre-trained large language models (LLMs) (Chowdhery et al., 2022; OpenAI, 2023; Wu et al., 2023) generalize well on unseen tasks by *few-shot prompting* (or *in-context learning (ICL)*) (Brown et al., 2020; Min et al., 2022; Chen et al., 2022; Li et al., 2023). This is performed by concatenating a few examples (e.g., question-answer pairs) as a prompt, and then appending the testing question. However, it is still challenging for LLMs to answer mathematical questions by simply prompting the question-answer pairs, as mathematics is more complex and often requires many steps to derive the answer.

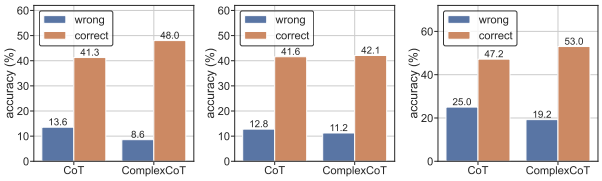


(a) *text-davinci-003*. (b) *GPT-3.5-Turbo*. (c) *GPT-4*.

Figure 1: Testing accuracy (averaged over six data sets) of Self-Consistency versus number of sampling paths (M_{F}).

Recently, Wei et al. (2022) propose *chain-of-thought (CoT) prompting* for LLMs, which generates explicit intermediate steps that are used to reach the answer. Specifically, each in-context example is augmented with several thinking steps described in natural language. A few examples are concatenated as a CoT prompt. In inference, the testing question is appended to the prompt and then fed to an LLM. The LLM is expected to imitate the in-context examples, i.e., generating several reasoning steps before giving the answer. CoT prompting has achieved promising performance on mathematical reasoning tasks (Wei et al., 2022; Wang et al., 2023; Zheng et al., 2023; Zhang et al., 2023b), and many works have been proposed to improve its effectiveness (Fu et al., 2023; Zheng et al., 2023; Zhou et al., 2023; Yao et al., 2023; Pitis et al., 2023) and efficiency (Zhang et al., 2023b; Kojima et al., 2022; Diao et al., 2023; Lu et al., 2022).

Self-Consistency (Wang et al., 2023) is a simple but effective approach to improve CoT prompting. Using temperature sampling (Ackley et al., 1985; Ficler and Goldberg, 2017), it samples a diverse set of reasoning chains which may lead to multiple candidate answers. The one that receives the most votes is then chosen as the final answer. Figure 1 shows the testing accuracy (averaged over six data sets) of Self-Consistency with different numbers (denoted by M_{F}) of sampling paths using three LLMs (the experimental setup is in Section 4.1). As can be seen, simply sampling more reasoning paths may not lead to performance improvement, particularly when M_{F} is large. Moreover, among



(a) *text-davinci-003*. (b) *GPT-3.5-Turbo*. (c) *GPT-4*.

Figure 2: Accuracy (averaged over all backward questions across the six data sets) of predicting the masked number in backward questions with correct/wrong candidate answers.

the failure questions of Self-Consistency, about 60% have at least one reasoning chain reaches the correct answer (Table 4 in Appendix A). Hence, the majority voting of Self-Consistency can be improved using a better verifier.

In this paper, we introduce *backward reasoning* (or *backward chaining*) (Pettit and Sugden, 1989; Russell and Norvig, 1995; Khot et al., 2021; Liang et al., 2021; Yu et al., 2023) for verifying candidate answers. While Self-Consistency uses *forward reasoning* for verification (i.e., starting with a question, the LLM generates multiple reasoning steps to reach its answer), backward reasoning works backward from a candidate answer to the antecedent for checking if any data supports this answer.

To use backward reasoning for verifying answers, we mask an informative word in the question and ask the LLM to predict the masked word when a candidate answer $\hat{A}_c$ is provided. We focus on mathematical reasoning tasks, where numbers are the informative words being masked. Extending backward reasoning to non-mathematical tasks is discussed in Section 3.4. For mathematical tasks, we mask a number in the question by "$\mathbf{x}$" and design a template "*If we know the answer to the above question is* $\{\hat{A}_c\}$*, what is the value of unknown variable* $\mathbf{x}$*?*" to form a backward question, which is then fed to the LLM to generate multiple steps before predicting the value of $\mathbf{x}$. As the ground-truth value of $\mathbf{x}$ is known, we can check whether the masked number is predicted exactly. Intuitively, a correct candidate answer is more likely to help predict the masked number than wrong answers (as verified in Figure 2). Unlike Self-Verification (Weng et al., 2023) which needs the assistance of an LLM to rewrite the question into a declarative statement (e.g., "How many hours does he spend on TV and reading in 4 weeks?" with a candidate answer of 36 is rewritten to "He spends 36 hours on TV and reading in 4 weeks"), we append the aforementioned simple template to the question without rewriting.

As forward reasoning and backward reasoning are complementary, we propose a FOrward-BAckward Reasoning (FOBAR) method to combine them. An illustration is shown in Figure 3. In the *forward* direction, we estimate $\mathbb{P}_{\text{F}}(\hat{A}_c)$, the probability that a candidate answer $\hat{A}_c$ is correct, as the proportion of votes $\hat{A}_c$ gets. In the *backward* direction, for each candidate $\hat{A}_c$, we create several questions for backward reasoning by masking numbers and sample a set of backward reasoning chains to predict each masked number. Then, by defining the vote of $\hat{A}_c$ as the number of chains that predict the masked number exactly, we estimate the probability $\mathbb{P}_{\text{B}}(\hat{A}_c)$ as the proportion of votes $\hat{A}_c$ gets in the backward direction. As a result, by combining backward and forward reasoning, we estimate the probability $\mathbb{P}(\hat{A}_c)$ as the geometric mean of forward and backward probabilities. Extensive experiments on six data sets and three OpenAI's LLMs (i.e., *text-davinci-003* (OpenAI, 2022a), *GPT-3.5-Turbo* (OpenAI, 2022b), and *GPT-4* (OpenAI, 2023)) show that FOBAR achieves state-of-the-art (SOTA) performance.

Our contributions are summarized as follows. (i) We introduce backward reasoning to mathematical verification by masking a number in the original question and asking the LLM to predict the masked number when a candidate answer is provided. (ii) We propose FOBAR to combine forward and backward reasoning for verification. (iii) Experimental results on six standard mathematical benchmarks and three LLMs show that FOBAR achieves SOTA performance. In particular, FOBAR outperforms Self-Consistency which uses forward reasoning alone, demonstrating that combining forward and backward reasoning together is better. Additionally, FOBAR outperforms Self-Verification, confirming that using the simple template and the proposed combination are more effective. (iv) Empirical results on two non-mathematical reasoning tasks show that FOBAR also performs well.

2 Related Work

Chain-of-Thought (CoT) Prompting. Wei et al. (2022) propose augmenting question-answer pairs with intermediate steps such that the LLM can solve questions step-by-step. Specifically, each in-context example is a triplet $(Q^{(i)}, R^{(i)}, A^{\star(i)})$, where $R^{(i)}$ is a reasoning chain with natural language descriptions of steps leading from the question $Q^{(i)}$ to the ground-truth answer $A^{\star(i)}$. In infer-

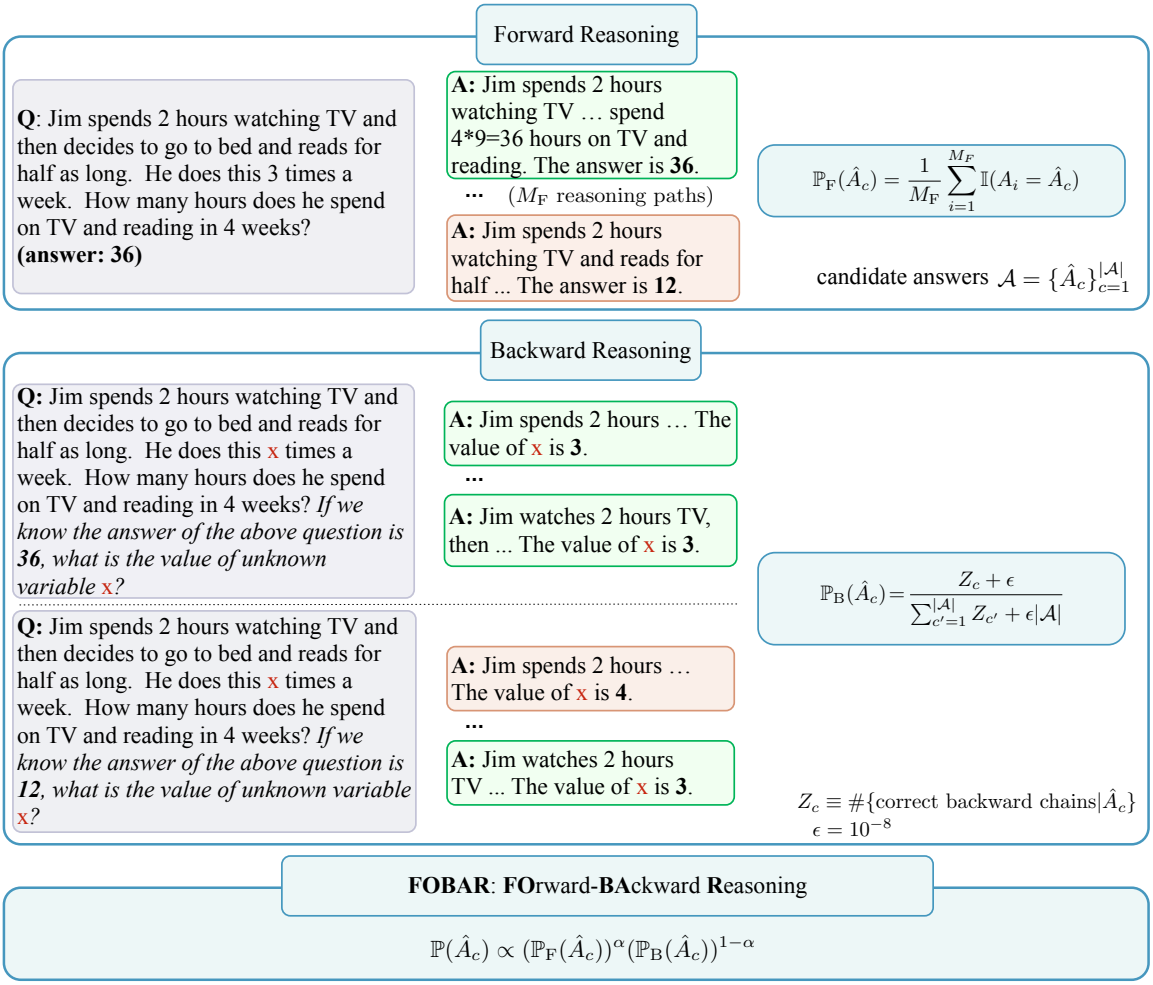


Figure 3: Overview of forward/backward reasoning and the proposed FOBAR. The detailed procedure is shown in Algorithm 1.

ence, a new question Q is appended to the prompt:

$$\mathbf{P}_{\text{CoT}} = \text{“Question: } Q^{(1)} \text{ \textbackslash n Answer: } R^{(1)}, A^{\star(1)} \ldots \text{ Question: } Q^{(K)} \text{ \textbackslash n Answer: } R^{(K)}, A^{\star(K)}\text{”}$$

and “$\mathbf{P}_{\text{CoT}}$ \n Question: Q \n Answer:” is fed to the LLM for generating both its reasoning chain R and answer A. CoT prompting has achieved SOTA performance on a wide variety of tasks (Wei et al., 2022; Kojima et al., 2022; Fu et al., 2023; Zhang et al., 2023b; Wang et al., 2023; Zheng et al., 2023; Zhou et al., 2023; Zhang et al., 2023c).

Recently, many works (Fu et al., 2023; Zheng et al., 2023; Madaan et al., 2023; Paul et al., 2023; Shinn et al., 2023; Welleck et al., 2023; Zhou et al., 2023; Chen et al., 2023; Zhang et al., 2023a) have been proposed to improve the quality of reasoning chains in CoT prompting. ComplexCoT (Fu et al., 2023) selects examples with more steps as in-context examples, while PHP (Zheng et al., 2023) iteratively uses the previous answers as hints in prompting. These methods can be viewed as *forward reasoning*, which starts from the question and generates a reasoning chain to reach the answer. Instead of taking a single reasoning chain by greedy decoding, Self-Consistency (Wang et al., 2023) samples a diverse set of chains and obtains a set of candidate answers. The final answer is then selected by majority voting.

Backward Reasoning. Backward reasoning (a.k.a. backward chaining) (Pettit and Sugden, 1989; Russell and Norvig, 1995; Khot et al., 2021; Liang et al., 2021; Yu et al., 2023) starts with an answer and works backward to verify the sequence of steps or conditions necessary to reach this answer. Backward reasoning is particularly useful in domains when the answer is known, e.g., in automated theorem provers (Russell and Norvig, 1995; Rocktäschel and Riedel, 2016; Wang and Deng, 2020; Kazemi et al., 2023; Poesia and Goodman, 2023). Recently, Self-Verification (Weng et al., 2023) rewrites the question with an answer into a declarative statement and then asks the LLM to predict the masked number. RCoT (Xue et al., 2023) regenerates a sentence (a sequence of tokens) in the question conditioning on the answer and detects whether there is factual inconsistency in the

constructed question by three complicated steps. The complicated checking procedure may lead to inaccurate verification. In contrast, for creating backward questions, we simply append a template to the original question without additional rewriting and reconstruction; for verification, the proposed FOBAR just needs to check whether the number is predicted correctly by string comparison, which is much simpler and more accurate. Furthermore, the proposed FOBAR combines forward and backward reasoning together for verification, while Self-Verification and RCoT use backward reasoning alone.

3 Forward-Backward Reasoning for Verification

In this section, we propose the FOBAR method for verification. An overview is shown in Figure 3. We first consider mathematical reasoning tasks. A set of candidate answers is generated in the forward direction, and we estimate each answer's probability based on the votes it receives (Section 3.1). Next, we mask a number in the question and propose a simple template to create backward questions for verifying candidate answers (Section 3.2). We further propose FOBAR (Section 3.3) to combine forward and backward reasoning. Extension to non-mathematical tasks is discussed in Section 3.4.

3.1 Forward Reasoning

Forward reasoning starts with a question and generates multiple intermediate steps toward the answer. Specifically, for a question Q, we prepend it with a base prompt $\mathbf{P}_{\mathrm{F}}$ (e.g., CoT prompting (Wei et al., 2022) or ComplexCoT prompting (Fu et al., 2023)) and feed the tuple $(\mathbf{P}_{\mathrm{F}}, Q)$ to the LLM for generating a reasoning chain and candidate answer. Using temperature sampling (Ackley et al., 1985; Ficler and Goldberg, 2017), we sample M_{F} candidate reasoning chains $\{R_i\}_{i=1}^{M_{\mathrm{F}}}$ and extract the corresponding candidate answers $\{A_i\}_{i=1}^{M_{\mathrm{F}}}$ (see Figure 3, top). Let $\mathcal{A} = \{\hat{A}_c\}_{c=1}^{|\mathcal{A}|}$ be the set of answers deduplicated from $\{A_i\}_{i=1}^{M_{\mathrm{F}}}$. Unlike greedy decoding (Wei et al., 2022), we may have several different candidate answers (i.e., $|\mathcal{A}| > 1$). We propose to estimate the probability that candidate $\hat{A}_c \in \mathcal{A}$ is correct as the proportion of votes it receives from the reasoning paths:

$$\mathbb{P}_{\mathrm{F}}(\hat{A}_c) = \frac{1}{M_{\mathrm{F}}} \sum_{i=1}^{M_{\mathrm{F}}} \mathbb{I}(A_i = \hat{A}_c), \qquad (1)$$

where $\mathbb{I}(\cdot)$ is the indicator function. Choosing $\hat{A}_c$ with the largest $\mathbb{P}_{\mathrm{F}}(\hat{A}_c)$ corresponds to the state-of-the-art method of Self-Consistency (Wang et al., 2023). However, as shown in Figure 1, **the performance of Self-Consistency saturates when M_{F} is sufficiently large.** Thus, simply sampling more reasoning paths brings negligible performance improvement.

3.2 Backward Reasoning

In backward reasoning, we mask a number contained in the question and ask the LLM to predict the masked number by using a provided candidate answer. Specifically, suppose that question Q involves N_Q numbers $\{\mathrm{num}^{(n)}\}_{n=1}^{N_Q}$. We replace each of them one by one with $\mathbf{x}$. The resultant masked question $\hat{Q}^{(n)}$ is then concatenated with the following template, which contains a candidate answer $\hat{A}_c \in \mathcal{A}$.

Template For Creating Backward Question

$\mathcal{T}(\hat{A}_c)$ = *If we know the answer to the above question is* $\{\hat{A}_c\}$, *what is the value of unknown variable* x*?*

Each $(\hat{Q}^{(n)}, \mathcal{T}(\hat{A}_c))$ pair is called a *backward question*. In total, we obtain N_Q backward questions. Some examples of backward questions are shown in Example B.1 of Appendix B. Note that Self-Verification (Weng et al., 2023) needs the assistance of an LLM to rewrite a (question, answer) pair into a declarative statement.[1] In contrast, the proposed template is simpler and avoids possible mistakes (an example illustrating Self-Verification's rewriting mistakes is shown in Appendix C).

To predict the masked number, we prepend the backward question with a prompt $\mathbf{P}_{\mathrm{B}}$, which consists of several (backward) question-answer demos with reasoning chains. An example question-answer demo is shown in Example B.2 of Appendix B. We feed each of $(\mathbf{P}_{\mathrm{B}}, \hat{Q}^{(n)}, \mathcal{T}(\hat{A}_c))$ (where $n = 1, \ldots, N_Q$) to the LLM, which then imitates the in-context examples in $\mathbf{P}_{\mathrm{B}}$ and generates a reasoning chain for the prediction of the masked number. We sample M_B such reasoning chains with predictions $\{\widehat{\mathrm{num}}_{c,b}^{(n)}\}_{b=1}^{M_{\mathrm{B}}}$ (see Figure 3, middle). For each candidate answer $\hat{A}_c$, we count the number of times that the masked number is exactly

[1] For example, "How many hours does he spend on TV and reading in 4 weeks?" with a candidate answer of 36 is rewritten to "He spends 36 hours on TV and reading in 4 weeks".

predicted:

$$Z_c = \sum_{n=1}^{N_Q} \sum_{b=1}^{M_\text{B}} \mathbb{I}(\widehat{\text{num}}_{c,b}^{(n)} = \text{num}^{(n)}). \quad (2)$$

The probability that candidate answer $\hat{A}_c$ is correct is estimated as

$$\mathbb{P}_\text{B}(\hat{A}_c) = \frac{Z_c + \epsilon}{\sum_{c'=1}^{|\mathcal{A}|} Z_{c'} + \epsilon|\mathcal{A}|}, \quad (3)$$

where $\epsilon = 10^{-8}$ is a small positive constant to avoid division by zero. One can simply choose $\hat{A}_c$ with the largest $\mathbb{P}_\text{B}(\hat{A}_c)$ as the prediction. A more effective method, as will be shown in Section 3.3, is to combine the probabilities obtained from forward and backward reasoning.

3.3 FOBAR (FOrward and BAckward Reasoning)

As forward and backward reasoning are complementary (i.e., backward reasoning may succeed in the cases where forward reasoning fails and vice versa, as Examples D.1 and D.2 provided in Appendix D), we propose to combine them for verification. Intuitively, a candidate answer is likely to be correct when it receives many votes in forward reasoning and also helps the LLM to predict the masked numbers in backward reasoning. We estimate the probability that $\hat{A}_c$ is correct as

$$\mathbb{P}(\hat{A}_c) \propto \big(\mathbb{P}_\text{F}(\hat{A}_c)\big)^{\alpha} \big(\mathbb{P}_\text{B}(\hat{A}_c)\big)^{1-\alpha}, \quad (4)$$

with weight $\alpha \in [0, 1]$ (see Figure 3, bottom). When $\alpha = 1$, it reduces to Self-Consistency (Wang et al., 2023); When α equals 0, it reduces to backward reasoning for verification. In the experiments, we combine the forward and backward probabilities by the geometric mean (i.e., $\alpha = 0.5$). Finally, we select the answer as $\arg\max_{\hat{A}_c \in \mathcal{A}} \mathbb{P}(\hat{A}_c)$. The whole procedure is shown in Algorithm 1.

Compared with training an LLM as verifier (Cobbe et al., 2021), which is computationally expensive and labor-intensive in collecting extra annotation data, FOBAR is training-free (thus, no additional data collection) and more effective in verification (Table 6 in Appendix E.1). The proposed backward reasoning can be combined with other forward reasoning methods such as step-by-step verification proposed by Ling et al. (2023) (Table 7 in Appendix E.2).

Algorithm 1 FOBAR.

Require: number of reasoning chains M_F and M_B, prompts $\mathbf{P}_\text{F}$ and $\mathbf{P}_\text{B}$; $\epsilon = 10^{-8}$; $\alpha = 0.5$;
Input: a question Q with N_Q numbers;
feed $(\mathbf{P}_\text{F}, Q)$ to LLM, sample M_F reasoning chains with candidate answers $\{A_i\}_{i=1}^{M_\text{F}}$;
deduplicate $\{A_i\}_{i=1}^{M_\text{F}}$ to $\mathcal{A} = \{\hat{A}_c\}_{c=1}^{|\mathcal{A}|}$;
compute $\mathbb{P}_\text{F}(\hat{A}_c)$ by Eq. (1) for $\hat{A}_c \in \mathcal{A}$;
for $\hat{A}_c \in \mathcal{A}$ **do**
  for $n = 1, \ldots, N_Q$ **do**
    create $\hat{Q}^{(n)}$ by masking the nth number $\text{num}^{(n)}$ in Q;
    feed $(\mathbf{P}_\text{B}, \hat{Q}^{(n)}, \mathcal{T}(\hat{A}_c))$ to LLM;
    sample M_B predictions $\{\widehat{\text{num}}_{c,b}^{(n)}\}_{b=1}^{M_\text{B}}$;
  end for
  compute Z_c by Eq. (2);
end for
compute $\mathbb{P}_\text{B}(\hat{A}_c)$ by Eq. (3) for $\hat{A}_c \in \mathcal{A}$;
compute $\mathbb{P}(\hat{A}_c)$ by Eq. (4) for $\hat{A}_c \in \mathcal{A}$;
return $\arg\max_{\hat{A}_c \in \mathcal{A}} \mathbb{P}(\hat{A}_c)$.

3.4 Extension to Non-Mathematical Tasks

In mathematical questions, numbers are the most informative words. For non-mathematical tasks, we can analogously mask an informative word and ask the LLM to guess the masked word given a candidate answer.

For example, consider the following question-answer pair from the *Last Letter Concatenation* task (Wei et al., 2022; Zhou et al., 2023): "Take the last letters of each word in '*Whitney Erika Tj Benito*' and concatenate them" with ground-truth answer "*yajo*". We can mask one of the four words (e.g., "*Erika*"). Given a candidate answer $\hat{A}_c$, we create a backward question as "Take the last letters of each word in '*Whitney* ___ *Tj Benito*' and concatenate them. If we know the answer to the above question is $\hat{A}_c$, which is the word at the blank, *Erika* or *Dqhjz*", where "*Dqhjz*" is obtained by shifting each letter of "*Erika*". The LLM is more likely to choose "*Erika*" if the second letter in $\hat{A}_c$ is "*a*".

4 Experiments

4.1 Setup

Datasets. Experiments are conducted on six benchmark mathematical data sets which are commonly used in evaluating CoT reasoning ability (Zheng et al., 2023; Wang et al., 2023):

(i) *AddSub* (Hosseini et al., 2014), (ii) *MultiArith* (Roy and Roth, 2015), (iii) *SingleEQ* (Koncel-Kedziorski et al., 2015), (iv) *SVAMP* (Patel et al., 2021), (v) *GSM8K* (Cobbe et al., 2021), (vi) *AQuA* (Ling et al., 2017). Some statistics and example question-answer pairs are shown in Table 8 in Appendix F. Questions in *AddSub* and *SingleEQ* are easier and do not need multi-step calculations. Questions in the other data sets are more challenging as many steps are required.

Baselines. We compare the proposed FOBAR with (i) In-Context Learning (ICL) using question-answer pairs as demonstrations (Brown et al., 2020), and recent CoT prompting methods, including: (ii) CoT prompting (Wei et al., 2022); (iii) ComplexCoT prompting (Fu et al., 2023) which selects demonstrations with complex reasoning steps; (iv) RE2 (Xu et al., 2023) which re-reads the question in the prompt; (v) PHP (Zheng et al., 2023) which iteratively uses the previous answers as hints in designing prompts; (vi) RCoT (Xue et al., 2023) which reconstructs the question based on the candidate answer and checks the factual inconsistency for verification; (vii) Self-Consistency (Wang et al., 2023), which samples multiple reasoning chains and selects the answer by majority voting; (viii) Self-Verification (Weng et al., 2023), which chooses the top-2 candidate answers obtained from Self-Consistency and re-ranks them based on the verification scores computed in the backward procedure.

Following Zheng et al. (2023), we experiment with three LLMs: (i) *text-davinci-003* (OpenAI, 2022a), (ii) *GPT-3.5-Turbo* (OpenAI, 2022b), and (iii) *GPT-4* (OpenAI, 2023). *GPT-3.5-Turbo* and *GPT-4* are more powerful than *text-davinci-003*. The proposed FOBAR is general and can be integrated into any prompting method. Here, we choose the CoT prompting and ComplexCoT prompting as base prompts as in Zheng et al. (2023).

Implementation Details. Following (Wang et al., 2023; Zhou et al., 2023; Zheng et al., 2023), the temperature for sampling is 0.7 for both forward and backward reasoning. The α in Eq. (4) is set to 0.5. For *text-davinci-003*, M_{F} is 40 as in (Wang et al., 2023; Zheng et al., 2023); whereas the more powerful LLMs (*GPT-3.5-Turbo* and *GPT-4*) use a smaller M_{F} (i.e., 10). M_{B} is set to 8 for all three LLMs. We do not repeat the experiments using different seeds as querying OpenAI's LLMs is costly.

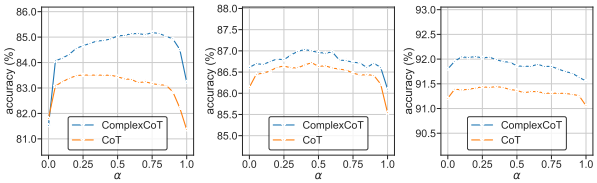
4.2 Results

Table 1 shows the testing accuracies. As can be seen, for all three LLMs, FOBAR with ComplexCoT prompting achieves the highest average accuracy. When using CoT as the base prompt, FOBAR outperforms Self-Consistency most of the time, demonstrating that combining forward and backward reasoning is better than using forward reasoning alone. Furthermore, FOBAR performs better than Self-Verification on almost all datasets, demonstrating that using the proposed simple template in backward reasoning and the proposed combination is more effective in verification. FOBAR (with either CoT or ComplexCoT) on *GPT-4* achieves the highest average accuracy, as *GPT-4* is currently the SOTA LLM. Moreover, for all three LLMs, FOBAR using ComplexCoT as base prompt achieves higher accuracy than using CoT on average, which is consistent with observations in (Fu et al., 2023; Zheng et al., 2023) that ComplexCoT is better than CoT.

4.3 Combining Forward and Backward Probabilities

In this experiment, we study how the combination weight α in Eq. (4) affects performance. Figure 4 shows the testing accuracies (averaged over the six data sets) with $\alpha \in [0, 1]$ using the three LLMs. As can be seen, FOBAR is insensitive to α over a large range for all three LLMs. In the sequel, we use $\alpha = 0.5$, which corresponds to the geometric mean of the forward and backward probabilities.

Alternatively, one can combine the forward and backward probabilities by the arithmetic mean, i.e., $\mathbb{P}(\hat{A}_c) = \frac{1}{2}\big(\mathbb{P}_{\text{F}}(\hat{A}_c) + \mathbb{P}_{\text{B}}(\hat{A}_c)\big)$. Figure 5 shows the testing accuracies for the three LLMs. As shown, the arithmetic mean has comparable performance as the geometric mean. Hence, Figures 4 and 5 together suggest that FOBAR is robust to the combination of forward and backward probabilities.



(a) *text-davinci-003*. (b) *GPT-3.5-Turbo*. (c) *GPT-4*.

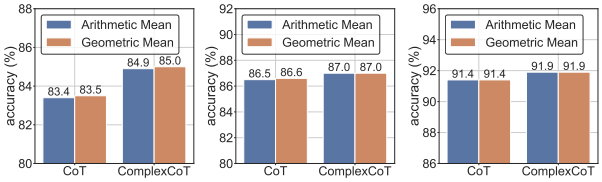
Figure 4: Testing accuracy (averaged over the six data sets) of FOBAR w.r.t. α.

4.4 Usefulness of Forward and Backward Reasoning

We perform an ablation study on forward (FO) and backward (BA) reasoning. We consider the

Table 1: Testing accuracies (%) on six data sets using three LLMs. For each LLM, methods are grouped according to the base prompt they used. The best in each group is in **bold**. Results with † are from the original publications. "–" means that the result is not reported in the original publication.

			AddSub	*MultiArith*	*SingleEQ*	*SVAMP*	*GSM8K*	*AQuA*	Average
text-davinci-003		ICL (Brown et al., 2020)	90.4	37.6	84.3	69.1	16.9	29.1	54.5
	CoT	CoT (Wei et al., 2022)	91.4	93.6	92.7	79.5	55.8	46.5	76.6
		PHP† (Zheng et al., 2023)	91.1	94.0	93.5	81.3	57.5	44.4	77.0
		RE2† (Xu et al., 2023)	91.7	93.3	93.3	81.0	61.6	44.5	77.6
		Self-Consistency (Wang et al., 2023)	91.7	95.9	94.5	83.1	67.9	**55.1**	81.4
		Self-Verification (Weng et al., 2023)	87.4	95.3	92.9	82.2	59.8	37.4	75.8
		FOBAR	**91.9**	**100.0**	**96.1**	**86.8**	**70.8**	**55.1**	**83.5**
	ComplexCoT	ComplexCoT (Fu et al., 2023)	88.9	95.3	93.7	78.0	67.7	48.8	78.7
		PHP† (Zheng et al., 2023)	**91.6**	96.6	95.0	83.7	68.4	53.1	81.4
		Self-Consistency (Wang et al., 2023)	89.4	98.5	91.1	82.7	**79.1**	**58.7**	83.2
		Self-Verification (Weng et al., 2023)	89.9	95.5	94.1	80.1	72.0	38.2	78.3
		FOBAR	90.6	**100.0**	**95.3**	**87.0**	78.7	**58.7**	**85.0**
GPT-3.5-Turbo		ICL (Brown et al., 2020)	88.6	87.6	88.8	80.6	32.2	31.1	68.2
	CoT	CoT (Wei et al., 2022)	89.4	97.9	92.9	84.2	77.2	54.3	82.7
		RE2† (Xu et al., 2023)	89.9	96.5	**95.3**	80.0	80.6	58.3	83.4
		Self-Consistency (Wang et al., 2023)	**90.6**	98.6	93.1	86.4	81.9	**62.6**	85.5
		Self-Verification (Weng et al., 2023)	90.4	97.4	92.9	83.1	74.9	60.6	83.2
		FOBAR	89.4	**99.3**	94.5	**88.9**	**85.1**	**62.6**	**86.6**
	ComplexCoT	Complex CoT (Fu et al., 2023)	87.9	98.3	**94.5**	81.1	80.7	59.1	83.6
		RCoT† (Xue et al., 2023)	88.2	–	93.0	84.9	84.6	53.3	–
		PHP† (Zheng et al., 2023)	85.3	98.0	92.9	83.1	85.1	60.6	84.2
		Self-Consistency (Wang et al., 2023)	88.1	98.8	**94.5**	85.0	86.4	63.0	86.0
		Self-Verification (Weng et al., 2023)	87.9	96.6	93.3	81.0	78.2	61.4	83.1
		FOBAR	**88.4**	**99.8**	94.3	**88.5**	**87.4**	**63.4**	**87.0**
GPT-4		ICL (Brown et al., 2020)	92.1	98.6	94.3	90.9	48.5	48.0	78.7
	CoT	CoT (Wei et al., 2022)	**92.7**	**99.0**	95.7	92.9	93.4	69.7	90.6
		Self-Consistency (Wang et al., 2023)	92.2	**99.0**	95.9	93.3	94.8	**71.3**	91.1
		Self-Verification (Weng et al., 2023)	**92.7**	**99.0**	95.7	93.1	93.7	70.1	90.7
		FOBAR	92.4	**99.0**	**96.1**	**94.1**	**95.4**	**71.3**	**91.4**
	ComplexCoT	Complex CoT (Fu et al., 2023)	**91.9**	98.3	94.5	92.4	95.1	72.4	90.8
		PHP† (Zheng et al., 2023)	89.6	98.1	93.1	91.9	95.5	**79.9**	91.3
		Self-Consistency (Wang et al., 2023)	91.4	98.5	**94.7**	93.4	96.2	75.2	91.6
		Self-Verification (Weng et al., 2023)	91.6	98.5	**94.7**	93.0	95.7	75.6	91.5
		FOBAR	**91.9**	**98.6**	**94.7**	**94.4**	**96.4**	75.2	**91.9**



(a) *text-davinci-003*. (b) *GPT-3.5-Turbo*. (c) *GPT-4*.

Figure 5: Testing accuracy of FOBAR (averaged over the six data sets) with geometric/arithmetic mean of forward and backward probabilities.

four combinations: (i) using neither forward nor backward reasoning (which reduces to greedy decoding (Wei et al., 2022)); (ii) use only forward reasoning (i.e., Self-Consistency); (iii) use only backward reasoning in selecting answers (i.e., $\alpha = 0$ in Algorithm 1); (iv) use both forward and backward reasoning (i.e., the proposed FOBAR). Table 2 shows the testing accuracies (averaged over the six data sets) for the three LLMs. As can be seen, in all the settings, using forward or backward reasoning is consistently better than using neither of them. Moreover, combining forward and backward reasoning is always the best. Examples D.1 and D.2 (Appendix D) show that FOBAR is able to rectify some failure cases of forward and backward reasoning, respectively.

Table 2: Average testing accuracies (%) with different combinations of forward (FO) and backward (BA) reasoning.

	FO	BA	*text-davinci-003*	*GPT-3.5-Turbo*	*GPT-4*
CoT	✗	✗	76.6	82.7	90.6
	✓	✗	81.4	85.5	91.1
	✗	✓	82.1	86.2	91.2
	✓	✓	**83.5**	**86.6**	**91.4**
ComplexCoT	✗	✗	78.7	83.6	90.8
	✓	✗	83.2	86.0	91.6
	✗	✓	81.3	86.3	91.8
	✓	✓	**85.0**	**87.0**	**91.9**

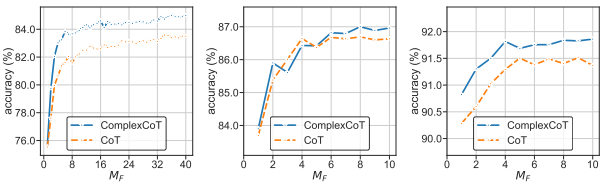
4.5 Correct Candidate Helps Backward Reasoning

In this experiment, we verify the intuition that the correct candidate answer helps LLM to predict the masked numbers. Figure 2 compares the accuracies of predicting the masked numbers in backward questions with the correct/wrong candidates. As can be seen, using the correct candidate has $2\times$ higher accuracy (averaged over the six data sets) than the wrong ones in predicting masked numbers, demonstrating that using backward reasoning for verifying candidate answers is reasonable.

4.6 Number of Forward and Backward Reasoning Chains

4.6.1 Varying $M_{\mathbf{F}}$

In this section, we study how the performance of FOBAR varies with the number of forward reasoning chains M_{F}. Figure 6 shows the testing accuracies (averaged over the six data sets) for the three LLMs. As can be seen, using a very small M_{F} (e.g., ≤ 5) is clearly undesirable, but the accuracy saturates quickly with increasing M_{F}. This suggests that one can use a small M_{F} to reduce the computational cost. Moreover, the accuracy curves of FOBAR are higher than those of Self-Consistency in Figure 1, again demonstrating that integrating backward reasoning into verification is effective.



(a) *text-davinci-003*. (b) *GPT-3.5-turbo*. (c) *GPT-4*.

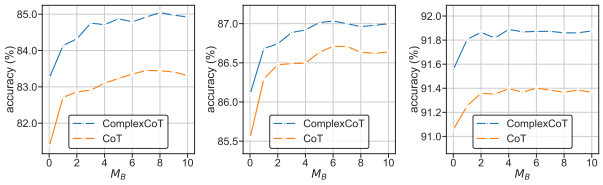
Figure 6: Testing accuracy of FOBAR (averaged over the six data sets) with M_{F}.

4.6.2 Varying $M_{\mathbf{B}}$

Next, we study how the performance of FOBAR varies with the number of backward reasoning chains M_{B}. Figure 7 shows the testing accuracies (averaged over the six data sets) for the three LLMs. Note that $M_{\mathrm{B}} = 0$ corresponds to using only forward reasoning. As shown, using a very small M_{B} (e.g., ≤ 4) is clearly undesirable, but the accuracy saturates quickly when M_{B} increases. Hence, using a small M_{B} can achieve a good balance between performance and efficiency.

4.7 Extension to Non-Mathematical Tasks

In this section, we perform experiments on two commonly-used non-mathematical tasks: *Date Understanding* (Wei et al., 2022; Fu et al., 2023) and *Last Letter Concatenation* (Wei et al., 2022; Zhou et al., 2023). Examples are shown in Table 8 (Appendix F. We compare FOBAR with other CoT-based methods and ICL using *GPT-3.5-Turbo*. PHP does not report results on non-mathematical tasks.



(a) *text-davinci-003*. (b) *GPT-3.5-turbo*. (c) *GPT-4*.

Figure 7: Testing accuracy of FOBAR (averaged over the six data sets) with M_{B}.

Table 3 shows the testing accuracies. As shown, FOBAR performs better than all the baselines with either CoT or ComplexCoT as base prompt. Moreover, all CoT-based methods significantly outperform ICL.

Table 3: Accuracies on the non-mathematical tasks of *Date Understanding* (denoted *DateU*) and *Last Letter Concatenation* (denoted *LastLetter*) using *GPT-3.5-Turbo*. Results with † are from the original publications. "–" means that the result is not reported in the original publication.

		DateU	*LastLetter*
	ICL (Brown et al., 2020)	52.0	8.0
CoT	CoT (Wei et al., 2022)	61.3	81.0
CoT	RE2† (Xu et al., 2023)	47.2	-
CoT	Self-Consistency (Wang et al., 2023)	65.6	81.4
CoT	Self-Verification (Weng et al., 2023)	66.1	81.8
CoT	FOBAR	**66.4**	**82.6**
ComplexCoT	ComplexCoT (Fu et al., 2023)	74.8	81.4
ComplexCoT	RCoT† (Xue et al., 2023)	71.7	-
ComplexCoT	Self-Consistency (Wang et al., 2023)	77.5	81.2
ComplexCoT	Self-Verification (Weng et al., 2023)	76.2	81.6
ComplexCoT	FOBAR	**78.0**	**82.4**

5 Conclusion

In this paper, we study the problem of verifying candidate answers to mathematical problems using chain-of-thought prompting. To complement the use of only forward reasoning for verification, we introduce backward reasoning: A simple template is introduced to create questions and a prompt is designed to ask the LLM to predict a masked word when a candidate answer is provided. Furthermore, we proposed FOBAR to combine forward and backward reasoning for verification. Extensive experiments on six standard mathematical data sets and three LLMs show that the proposed FOBAR achieves state-of-the-art performance on mathematical reasoning tasks. FOBAR can also be used on non-mathematical tasks and achieves superior performance.

6 Limitations and Potential Risks

Limitations In this paper, we focused on mathematical reasoning tasks, with extension to two non-mathematical reasoning tasks. However, extensions to more complicated non-mathematical reasoning tasks such as Common-Sense Question-Answering (CSQA) (Wei et al., 2022) and StrategyQA (Wei et al., 2022; Fu et al., 2023) are still to be explored, as identifying the informative words to mask is more challenging.

Potential Risks All data sets used in this work do not contain any information that names or uniquely identifies individual people or offensive content. Hence, there is no concern about ethical considerations and data privacy.

References

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. 1985. A learning algorithm for Boltzmann machines. *Cognitive Science*.

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

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to Self-Debug. Preprint arXiv:2304.05128.

Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2022. Meta-learning via language model in-context tuning. In *Annual Meeting of the Association for Computational Linguistics*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling language modeling with pathways. Preprint arXiv:2204.02311.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Hesse Christopher, and Schulman John. 2021. Training verifiers to solve math word problems. Preprint arXiv:2110.14168.

Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. 2023. Active prompting with chain-of-thought for large language models. Preprint arXiv:2302.12246.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Workshop on Stylistic Variation*.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning. In *International Conference on Learning Representations*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Conference on Empirical Methods in Natural Language Processing*.

Seyed Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. 2023. LAMBADA: Backward chaining for automated reasoning in natural language. In *Annual Meeting of the Association for Computational Linguistics*.

Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2021. Text modular networks: Learning to decompose tasks in the language of existing models. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Neural Information Processing Systems*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*.

Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. 2023. DeepInception: Hypnotize large language model to be jailbreaker. Preprint arXiv:2311.03191.

Zhengzhong Liang, Steven Bethard, and Mihai Surdeanu. 2021. Explainable multi-hop verbal reasoning through internal monologue. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. Preprint arXiv:2305.20050.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Annual Meeting of the Association for Computational Linguistics*.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of Chain-of-Thought reasoning. In *Neural Information Processing Systems*.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2022. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *International Conference on Learning Representations*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative refinement with self-feedback. In *Neural Information Processing Systems*.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022. MetaICL: Learning to learn in context. In *North American Chapter of the Association for Computational Linguistics*.

OpenAI. 2022a. GPT-3.5. Technical Report.

OpenAI. 2022b. Introducing ChatGPT. Technical Report.

OpenAI. 2023. GPT-4. Technical Report.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2023. REFINER: Reasoning feedback on intermediate representations. Preprint arXiv:2304.01904.

Philip Pettit and Robert Sugden. 1989. The backward induction paradox. *The Journal of Philosophy*.

Silviu Pitis, Michael R Zhang, Andrew Wang, and Jimmy Ba. 2023. Boosted prompt ensembles for large language models. Preprint arXiv:2304.05970.

Gabriel Poesia and Noah D Goodman. 2023. Peano: learning formal mathematical reasoning. *Philosophical Transactions of the Royal Society A*.

Tim Rocktäschel and Sebastian Riedel. 2016. Learning knowledge base inference with neural theorem provers. In *Workshop on Automated Knowledge Base Construction*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Empirical Methods in Natural Language Processing*.

Staurt J Russell and Peter Norvig. 1995. *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Neural Information Processing Systems*.

Mingzhe Wang and Jia Deng. 2020. Learning to prove theorems by learning to generate theorems. In *Neural Information Processing Systems*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Neural Information Processing Systems*.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2023. Generating sequences by learning to Self-Correct. In *International Conference on Learning Representations*.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with Self-Verification. In *Empirical Methods in Natural Language Processing*.

Zhenyu Wu, YaoXiang Wang, Jiacheng Ye, Jiangtao Feng, Jingjing Xu, Yu Qiao, and Zhiyong Wu. 2023. OpenICL: An open-source framework for in-context learning. In *Annual Meeting of the Association for Computational Linguistics*.

Xiaohan Xu, Chongyang Tao, Tao Shen, Can Xu, Hongbo Xu, Guodong Long, and Jian-guang Lou. 2023. Re-Reading improves reasoning in language models. Preprint arXiv:2309.06275.

Tianci Xue, Ziqi Wang, Zhenhailong Wang, Chi Han, Pengfei Yu, and Heng Ji. 2023. RCOT: Detecting and rectifying factual inconsistency in reasoning by reversing chain-of-thought. Preprint arXiv:2305.11499.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate problem solving with large language models. In *Neural Information Processing Systems*.

Fei Yu, Hongbo Zhang, and Benyou Wang. 2023. Nature language reasoning: A survey. Preprint arXiv:2303.14725.

Kechi Zhang, Zhuo Li, Jia Li, Ge Li, and Zhi Jin. 2023a. Self-Edit: Fault-aware code editor for code generation. In *Annual Meeting of the Association for Computational Linguistics*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023b. Automatic chain of thought prompting in large language models. In *International Conference on Learning Representations*.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. 2023c. Multimodal chain-of-thought reasoning in language models. In *International Conference on Machine Learning*.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023. Progressive-hint prompting improves reasoning in large language models. Preprint arXiv: 2304.09797.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*.

A Failure Cases of Self-Consistency and FOBAR

We conduct an analysis on the failure cases of Self-Consistency and FOBAR on the six data sets, using *GPT-3.5-Turbo* with ComplexCoT prompting. Table 4 shows the number of failure cases of Self-Consistency, with a breakdown into the numbers of cases with no chain reaching the correct answer and at least one chain reaching the correct answer. As can be seen, about 60% of the total failure cases have at least one correct chains (the remaining 40% have no correct chains and thus cannot be solved by backward reasoning). These 60% cases can potentially be fixed with a better verifier (such as the proposed FOBAR). Table 5 shows the statistics on the failure cases of FOBAR. As can be seen, FOBAR rectifies 54 (i.e., $294 - 240$) out of the 294 failure cases that have at least one correct answer in Self-Consistency.

Table 4: Statistics on the failure cases of Self-Consistency on the six data sets.

	AddSub	*MultiArith*	*SingleEQ*	*SVAMP*	*GSM8K*	*AQuA*	Total
#failures	47	7	28	150	179	94	505
#failures with no correct answer	28	0	14	57	60	52	211
#failures with at least one correct answer	19	7	14	93	119	42	294

Table 5: Statistics on the failure cases of FOBAR on the six data sets.

	AddSub	MultiArith	SingleEQ	SVAMP	GSM8K	AQuA	Total
#failures	46	1	29	115	166	94	451
##failures with no correct answer	28	0	14	57	60	52	211
#failures with at least one correct answer	18	1	15	58	106	42	240

B Question-Answer Demos of Backward Reasoning

Example B.1 shows three backward questions that mask different numbers in the original question. Example B.2 shows a backward question and its answer.

Example B.1: Backward questions.

Question: Jim spends x hours watching TV and then decides to go to bed and reads for half as long. He does this 3 times a week. How many hours does he spend on TV and reading in 4 weeks? *If we know the answer to the above question is* $\{\hat{A}_c\}$*, what is the value of unknown variable* x*?*
Question: Jim spends 2 hours watching TV and then decides to go to bed and reads for half as long. He does this x times a week. How many hours does he spend on TV and reading in 4 weeks? *If we know the answer to the above question is* $\{\hat{A}_c\}$*, what is the value of unknown variable* x*?*
Question: Jim spends 2 hours watching TV and then decides to go to bed and reads for half as long. He does this 3 times a week. How many hours does he spend on TV and reading in x weeks? *If we know the answer to the above question is* $\{\hat{A}_c\}$*, what is the value of unknown variable* x*?*

Example B.2: Backward Reasoning.

Question: Randy has 60 mango trees on his farm. He also has x less than half as many coconut trees as mango trees. How many trees does Randy have in all on his farm? *If we know the answer to the above question is 85, what is the value of unknown variable* x*?*
Answer: Let's think step by step. We know that Randy has 60 mango trees on his farm. We also know that he has x less than half as many coconut trees as mango trees. Let's use C to represent the number of coconut trees. So we can write: C = (1/2)*60 - x = 30 - x. The total number of trees on Randy's farm is the sum of the number of mango trees and coconut trees: 60 + (30 - x) = 90 - x. We are given that the total number of trees on Randy's farm is 85, so we can write: 90 - x = 85. Solving for x, we get: x = 5. The value of x is 5.

C Example Rewriting Mistake in Self-Verification

Question: A class of 50 students has various hobbies. 10 like to bake, 5 like to play basketball, and the rest like to either play video games or play music. How many like to play video games if the number that like to play music is twice the number that prefer playing basketball? (answer: 25)

We mask the first number (i.e., 50) by $\mathbf{x}$ and a candidate answer 25 is provided. The following shows the backward questions obtained by Self-Verification and FOBAR. We can see that Self-Verification makes a mistake in rewriting the question into a declarative statement, while the proposed simple template in FOBAR does not need rewriting.

Question (Self-Verification): A class of x students has various hobbies. 10 like to bake, 5 like to play basketball, and the rest like to either play video games or play music. The number of people who like to play video games is equal to the number of people who prefer playing basketball multiplied by two. The number of people who like to play video games is 25. What is the answer of x?
Question (FOBAR): A class of x students has various hobbies. 10 like to bake, 5 like to play basketball, and the rest like to either play video games or play music. How many like to play video games if the number that like to play music is twice the number that prefer playing basketball? *If we know the answer to the above question is 25, what is the value of unknown variable* x*?*

D Example Cases showing that Forward and Backward Reasoning are Complementary

In this section, we show that forward and backward reasoning are complementary, i.e., failure cases in forward reasoning can be corrected by backward reasoning, and vice versa. We use cases from the *SingleEQ* data set using *text-davinci-003* with CoT prompting. Example D.1 shows a case where forward reasoning (i.e., Self-Consistency) fails but backward reasoning succeeds. We can see that this problem is difficult to solve in the forward direction, but the correctness of a candidate answer can be easily verified in the backward direction. Example D.2 shows a case where backward reasoning fails but forward reasoning succeeds. Moreover, FOBAR can choose the correct answer in both cases.

Example D.1: Forward reasoning fails but backward reasoning succeeds.

Question: The sum of three consecutive odd numbers is 69. What is the smallest of the three numbers?
Ground-truth answer: 21
Forward reasoning: $\mathbb{P}_{\text{F}}(21) = 0.4, \mathbb{P}_{\text{F}}(23) = 0.6$
Backward reasoning: $\mathbb{P}_{\text{B}}(21) = 0.8, \mathbb{P}_{\text{B}}(23) = 0.2$
FOBAR: $\mathbb{P}(21) = 0.62, \mathbb{P}(23) = 0.38$
A backward question: The sum of three consecutive odd numbers is $\mathbf{x}$. What is the smallest of the three numbers? If we know the answer to the above question is 21, what is the value of unknown variable $\mathbf{x}$?

Example D.2: Forward reasoning succeeds but backward reasoning fails.

Question: While digging through her clothes for ice cream money, Joan found 15 dimes in her jacket, and 4 dimes in her shorts. How much money did Joan find?
Ground-Truth answer: 1.9
Forward reasoning: $\mathbb{P}_{\text{F}}(1.9) = 0.7, \mathbb{P}_{\text{F}}(190) = 0.3$
Backward reasoning: $\mathbb{P}_{\text{B}}(1.9) = 0.43, \mathbb{P}_{\text{B}}(190) = 0.57$
FOBAR: $\mathbb{P}(1.9) = 0.57, \mathbb{P}(190) = 0.43$
A backward question: While digging through her clothes for ice cream money, Joan found 15 dimes in her jacket, and $\mathbf{x}$ dimes in her shorts. How much money did Joan find? If we know the answer to the above question is 1.9, what is the value of unknown variable $\mathbf{x}$?

E Additional Experiments

E.1 Comparison between FOBAR and Trained Verifiers

Compared with Cobbe et al. (2021), which trains an LLM for verifying answers, FOBAR has two advantages. (i) **(training-free)** Training an LLM for verification is computationally expensive and labor-intensive in collecting extra annotation data, while backward reasoning for verification is training-free and requires no additional data collection. (ii) **(more effective)** As training the GPT-3 (175B) model is extremely expensive and their code is not publicly available, we compare our FOBAR with the result reported in Figure 5 of (Cobbe et al., 2021), where the candidate answers are generated by *GPT-3*. Table 6 shows the accuracy of *GSM8K*. As shown, FOBAR consistently performs much better than the trained verifier (**+14.8**).

Table 6: Comparison between FOBAR and a trained verifier on *GSM8K*.

Training GPT-3 (175B) for Verification (Cobbe et al., 2021)	56.0
FOBAR (text-davinci-003 + CoT)	70.8
FOBAR (text-davinci-003 + ComplexCoT)	78.7
FOBAR (GPT-3.5-Turbo + CoT)	85.1
FOBAR (GPT-3.5-Turbo + ComplexCoT)	87.4
FOBAR (GPT-4 + CoT)	95.4
FOBAR (GPT-4 + ComplexCoT)	96.4

E.2 Comparison between FOBAR and Step-by-Step Forward Verification

Recent works (Lightman et al., 2023; Ling et al., 2023) propose verifying the steps of forward reasoning chains. Lightman et al. (2023) propose to label exclusively steps of forward reasoning chains generated by LLMs. The labeled data are then used to train an LLM for verification. Compared with (Lightman et al., 2023), which is computationally expensive in training an LLM and labor-intensive in labeling data, our backward reasoning is training-free for verification and requires no additional data annotation.

Ling et al. (2023) propose a natural language-based deductive reasoning format that allows the LLM to verify **forward** reasoning steps. Different from (Ling et al., 2023), we use **backward** reasoning to verify the candidate answers instead of the steps in forward chains. As backward and forward reasoning are complementary, the proposed backward reasoning can be combined with their step-by-step forward methods. We replace the forward reasoning in FOBAR (i.e., Eq. (4)) with step-by-step verification proposed by Ling et al. (2023), and conduct experiments on *AddSub*, *GSM8K*, and *AQuA* using *GPT-3.5-Turbo*. Table 7 shows the testing accuracy. As can be seen, combining backward reasoning with forward reasoning methods consistently boosts performance.

Table 7: Accuracy of FOBAR when combining backward reasoning with three types of forward reasoning for verification.

	AddSub	*GSM8K*	*AQuA*
Self-Consistency	88.1	86.4	63.0
Self-Consistency + Backward Reasoning	**88.4**	**87.4**	**63.4**
NP (Ling et al., 2023)	93.67	87.05	70.34
NP + Backward Reasoning	**93.92**	**87.89**	**71.65**
NP + Deductive Verification + UPV (Ling et al., 2023)	93.54	86.01	69.49
NP + Deductive Verification + UPV + Backward Reasoning	**93.92**	**87.19**	**70.86**

F Data Sets

Table 8 shows the statistics on the data sets used in the experiments.

Table 8: Statistics of data sets used in the experiments.

		#samples	N_Q (mean $\pm$ std)	example
Math	*AddSub*	395	2.6 ± 0.7	Benny picked 2 apples and Dan picked 9 apples from the apple tree. How many apples were picked in total?
	MultiArith	600	3.1 ± 0.3	Katie picked 3 tulips and 9 roses to make flower bouquets. If she only used 10 of the flowers though, how many extra flowers did Katie pick?
	SingleEQ	508	2.2 ± 0.7	Joan went to 4 football games this year. She went to 9 football games last year. How many football games did Joan go to in all?
	SVAMP	1000	2.8 ± 0.7	Rachel has 4 apple trees. She picked 7 apples from each of her trees. Now the trees have a total 29 apples still on them. How many apples did Rachel pick in all?
	GSM8K	1319	3.8 ± 1.6	A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?
	AQuA	254	2.9 ± 1.3	If the population of a city increases by 5% annually, what will be the population of the city in 2 years time if its current population is 78000? Answer Choices: (A) 81900 (B) 85995 (C) 85800 (D) 90000 (E) None of these
Non-Math	*Last Letter Concatenation*	500	4.0 ± 0.0	Take the last letters of each word in "*Whitney Erika Tj Benito*" and concatenate them.
	Date Understanding	369	1.2 ± 0.7	The deadline is Jun 1, 2021, which is 2 days away from now. What is the date a month ago in MM/DD/YYYY?