Solving for X and Beyond: Can Large Language Models Solve Complex Math Problems with More-Than-Two Unknowns?

Kuei-Chun Kao[1], **Ruochen Wang**[1], **Cho-Jui Hsieh**[1]
[1]Department of Computer Science, University of California, Los Angeles
{johnson0213, ruocwang}@g.ucla.edu
chohsieh@cs.ucla.edu

https://johnsonkao0213.github.io/Formulate_and_Solve/

Abstract

Large Language Models (LLMs) have demonstrated remarkable performance in solving math problems, a hallmark of human intelligence. Despite high success rates on current benchmarks; however, these often feature simple problems with only one or two unknowns, which do not sufficiently challenge their reasoning capacities. This paper introduces a novel benchmark, BeyondX, designed to address these limitations by incorporating problems with multiple unknowns. Recognizing the challenges in proposing multi-unknown problems from scratch, we developed BeyondX using an innovative automated pipeline that progressively increases complexity by expanding the number of unknowns in simpler problems. Empirical study on BeyondX reveals that the performance of existing LLMs, even those fine-tuned specifically on math tasks, significantly decreases as the number of unknowns increases - with a performance drop of up to 70% observed in GPT-4. To tackle these challenges, we propose the Formulate-and-Solve strategy, a generalized prompting approach that effectively handles problems with an arbitrary number of unknowns. Our findings reveal that this strategy not only enhances LLM performance on the BeyondX benchmark but also provides deeper insights into the computational limits of LLMs when faced with more complex mathematical challenges.

1 Introduction

Mathematical problem-solving is a fundamental aspect of human intelligence, necessitating both language comprehension and reasoning skills. Recently, LLMs pretrained on extensive web-scale datasets, have exhibited exceptional abilities in addressing a variety of complex tasks. Consequently, mathematical challenges are frequently employed to benchmark the reasoning abilities of LLMs. Studies have shown that these models demonstrate human-level efficacy in solving math problems, aided by diverse prompting techniques include in-context learning (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2022) and the integration of external computational tools (Gao et al., 2022; Chen et al., 2022; Liu et al., 2023; He-Yueya et al., 2023).

Existing math datasets (see Table 7) used to evaluate LLMs often consist of algebraic problems involving only one or two unknown variables. While current results on these datasets are promising, their simplicity masks the true capabilities and limitations of these models. For instance, GPT-4 (Achiam et al., 2023) achieves a 98% success rate on the GMS8K (Cobbe et al., 2021) dataset, suggesting that performance on these benchmarks is nearing saturation. This highlights the need for the development of more complex problem sets designed to rigorously stress test LLMs and provide a more accurate measure of their performance.

While quantifying the complexity of these math problems is multi-dimensional, one common measure is the number of unknowns required to solve the problem. Problems with more unknowns involve larger systems of equations, reflecting more complex relationships between the quantities, and thus demanding more sophisticated solving methods. However, creating datasets that include problems with multiple unknowns presents significant challenges, as it is difficult for humans to manually develop a sufficient number of these complex problems from scratch. As a result, **existing math datasets are dominated by problems with at most two unknowns** (Cobbe et al., 2021; Koncel-Kedziorski et al., 2016, 2015; Roy and Roth, 2018).

This paper tackles the aforementioned challenge systematically, by presenting three key contributions: (1) the development of a multi-unknown math benchmark, (2) an empirical study assessing the performance of current LLMs on this new benchmark, and (3) the introduction of a specialized prompting strategy designed to enhance the ability of LLMs to solve multi-unknown problems.

Findings of the Association for Computational Linguistics: EACL 2024, pages 16821–16843
November 12-16, 2024 ©2024 Association for Computational Linguistics

C1: BeyondX - The first multi-unknown algebraic benchmark. To efficiently generate a large corpus of multi-unknown problems, we developed a novel pipeline that automatically expands existing problems to N unknowns. This pipeline operates on three key ideas: (1). *Scenario Expansion*: New problems are derived by extending the scenarios (such as financial calculations or grocery shopping) of existing simpler problems, ensuring contextual relevance. (2). *Progressive Extrapolation*: We add unknowns incrementally — one at a time — progressing from problems with two unknowns to three, four, and so on. This step-by-step approach makes the problem generation process significantly more manageable. (3). *Decomposed Problem Generation*: Instead of creating an entire problem at once, we decompose the process. The LLM is carefully instructed to first introduce new unknown variables into the base scenarios, generate the corresponding equations, translate these equations into semantic statements, and finally integrate them into the comprehensive problem statement.

C2: Existing LLMs struggles with increasing unknowns. Utilizing our BeyondX benchmark, we conduct a comprehensive evaluation of current LLMs, which includes both general-purpose models like GPT-3.5 (Brown et al., 2020), GPT-4, Gemini-Pro (Team et al., 2023), and Mistral-7B (Jiang et al., 2023) as well as models specifically fine-tuned on mathematical problems (WizardMath (Luo et al., 2023), OpenMath (Toshniwal et al., 2024), and MetaMath (Yu et al., 2023)). Our findings reveal a significant drop in performance as the number of unknowns in problems increases - a staggering $\sim 70\%$ degradation on GPT-4 for instance.

- This marked decline indicates that current benchmarks may overstate the algebraic capabilities of these models.
- Additionally, despite efforts to fine-tune LLMs on previous math corpus, they still struggle with multi-unknown problems.
- Even sophisticated prompting strategies, which utilize detailed natural language explanations, fail to substantially aid LLMs in overcoming these more complex challenges.

C3: Formulate-and-Solve A prompting method to tackle multi-unknown problems. Traditional prompting methods for LLMs typically do not account for the complexity inherent in systems of equations, potentially limiting the math-solving capabilities of these models. Addressing whether the observed performance drop in LLMs is primarily due to inadequate prompting strategies forms a critical part of future investigation. As the initial step, we propose Formulate-and-Solve, an automated prompting method that generalizes to an arbitrary number of unknowns. This strategy refines current approaches by integrating general math-solving principles to automatically craft relevant multi-unknown in-context examples for LLMs.

Our empirical evaluations demonstrate that Formulate-and-Solve outperforms traditional prompting methods on both standard algebra problem datasets and our more challenging BeyondX dataset. Importantly, our findings suggest that while the inherent limitations of LLMs contribute to their underperformance on complex problems, inadequate prompting strategies are a substantial bottleneck. By enhancing these strategies, Formulate-and-Solve not only improves LLM performance but also provides clearer insights into the actual computational limitations of current models when faced with advanced mathematical challenges.

2 Related Work

2.1 Math Word Problem Generation

Early research on math word problem (MWP) generation relied heavily on pre-defined structures, including domain knowledge, equations, and text templates (Nandhini and Balasundaram, 2011; Williams, 2011; Polozov et al., 2015). More recently, researchers began using pre-trained models fine-tuned on equation-to-MWP examples (Wang et al., 2021). Studies on using LLMs for MWP generation are scarce. Existing work includes evaluating GPT-3's ability to mimic specific problem types (Zong and Krishnamachari, 2023) and using GPT-4 to improve readability in existing problems (Norberg et al., 2023). However, these approaches are limited to replicating existing problem structures, such as the number of unknowns or equation templates. Our work focuses on how to expand existing one or two unknown problems into more complex multiple unknowns.

2.2 Math Word Problem Solver

Mathematical reasoning skills are crucial for intelligent systems, leading to a surge in research. In the past, studies focused on how statistical and deep learning NLP models could solve arithmetic and algebraic problems (Hosseini et al., 2014; Koncel-Kedziorski et al., 2015; Roy and Roth, 2016; Liang et al., 2016; Ling et al., 2017). Recently, researchers have introduced increasingly challenging math datasets (Saxton et al., 2019; Amini et al., 2019; Miao et al., 2020; Patel et al., 2021; Hendrycks et al., 2021) to improve difficulty, diversity, and robustness. However, these methods struggle to adapt to problems outside their training data. This limitation has driven the rise of LLMs in mathematical reasoning.

2.3 Math Reasoning with LLMs

Many prompting techniques have emerged to unlock the reasoning abilities of LLMs (Qiao et al., 2022). Chain-of-Thought (CoT) Prompting (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2022) was proposed to generate the reasoning steps before submitting the answer. Later, several other works (Nye et al., 2021; Zhou et al., 2022; Drozdov et al., 2022; Wang et al., 2023) also proposed different approaches to utilize LLMs to solve reasoning tasks by allowing intermediate steps or planning first before solving. All of these methods allow LLMs to process all steps without using any external tools or refinements. For incorporating external tools, Programming-of-Thought (PoT) prompting (Chen et al., 2022; Gao et al., 2022) utilizes LLMs with code pretraining to write a program as a rationale that explains the reasoning process. Equation-of-Thought (EoT) (Liu et al., 2023; He-Yueya et al., 2023) prompting tackles MWPs by converting them into linear equation systems, which are then solved by a symbolic solver. Although PoT and EoT tried to use external tools to solve MWPs, they did not consider the scenario of multiple unknown variables.

Recent LLMs advancements for math reasoning involve various training approaches. One method focuses on pretraining data specifically designed for math, such as Minerva (Lewkowycz et al., 2022), Llemma (Azerbayev et al., 2023), and DeepSeekMath (Shao et al., 2024). Another approach involves generating synthetic questions and answers that mimic existing benchmarks. For example, the WizardMath series (Luo et al., 2023) that improves mathematical reasoning in Mistral 7B (Jiang et al., 2023) with problems sourced primarily from GSM8K and MATH (Hendrycks et al., 2021) via output from closed-source LLMs. MetaMath (Yu et al., 2023) and MMIQC (Liu and Yao, 2024) focus on expanding existing questions in GSM8K and MATH. MetaMath rewrites questions in various ways, while MMIQC combines existing math pretraining data such as OpenWebMath (Paster et al., 2023) with question-answer variations from MetaMath. The Mammoth series (including Mammoth2) (Yue et al., 2023, 2024) uses curated instruction tuning datasets (MathInstruct, WebInstrcut) with reasoning rationales for training. The OpenMathInstruct (Toshniwal et al., 2024) series utilizes synthetic instruction data from open-source LLMs with strong math reasoning abilities.

3 Automatic Generation of Multi-Unknown Algebra Problems via Progressive Expansion

3.1 Challenges for Constructing Multi-Unknown Datasets

Generating new problems with LLMs. Creating correct, diverse, and solvable math problems manually is an exceptionally laborious task. The complexity of this task increases with the addition of each unknown, as more unknowns require consideration of additional relationships within the problem scenario. To automatize this process, we employ LLMs to generate the problems, with human verifiers subsequently ensuring the quality and solvability of these problems.

Limitations of naive generation. Directly prompting LLMs to generate multi-unknown algebra problems has often resulted in poor quality outputs. Firstly, generating problem scenarios from scratch tends to produce a narrow range of problem types, as evidenced by the lack of diversity reported in Table 18. Secondly, attempting to generate all relevant relationships and corresponding equations in a single step frequently leads to violations of problem constraints, rendering many problems unsolvable as detailed in Table 19.

3.2 Generating New Problems via Progress Expansion

Pipeline overview. To address the aforementioned challenges, we introduce a novel approach called Progressive Expansion, which applies a divide-and-conquer strategy: (1). *Scenario Diversification:* We begin by expanding existing simpler problems to increase scenario diversity. This leverages the rich variety of simpler problem scenarios as a foundation for more complex questions. (2).

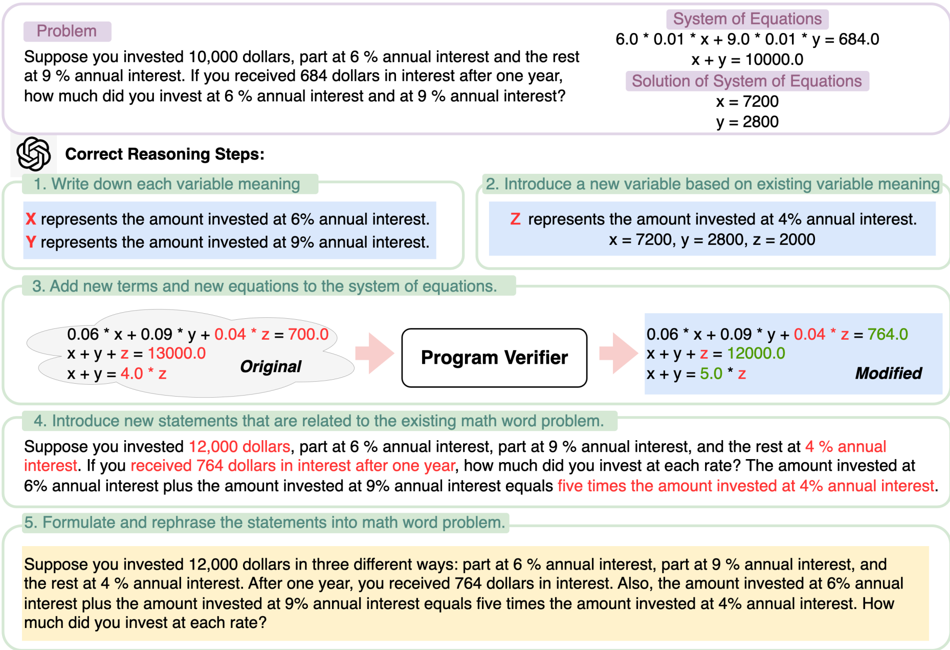


Figure 1: An example question of multi-unknown algebra problem generation and its corresponding reasoning steps. The prompts used for each step can be found Appendix 16.

Incremental Expansion: Instead of expanding problems from 1-2 unknowns to N unknowns in a single leap, we incrementally introduce one new variable at a time. This step-by-step approach simplifies the transformation from $N-1$ unknowns to N, making it more manageable and controllable for LLMs. (3). *Enhanced Solvability:* The problem expansion is broken into several simpler stages, making the entire generation more tractable for LLMs.

Multi-step problem expansion. The process of expanding problems is systematically divided into five steps, as illustrated in Figure 1. *Step 1: Understanding the source problem.* First, we instruct the LLM to analyze the original problem (including solutions) and explicitly explain the role of each unknown variable. This “perception step" lays the groundwork for subsequent expansions. *Step 2: Introducing a new unknown.* The LLM then introduces an additional unknown variable related to the existing problem framework and assigns an appropriate initial value (oracle value) to this variable. *Step 3: Expanding equation sets.* Next, the LLM generates a new equation that delineates the quantitative relationships between the new and existing variables. To ensure these equations are solvable, we integrate a Program Verifier module to assess and adjust their correctness as needed. *Step 4: Add equations to the problem statement* We translate the newly formed equations into text and incorporate them into the original problem statement to maintain consistency and flow. *Step 5: Final refinement.* Finally, we engage the LLM in a thorough polishing phase to refine the problem statement, ensuring it is fluent and coherent.

3.3 Constructing the Benchmark

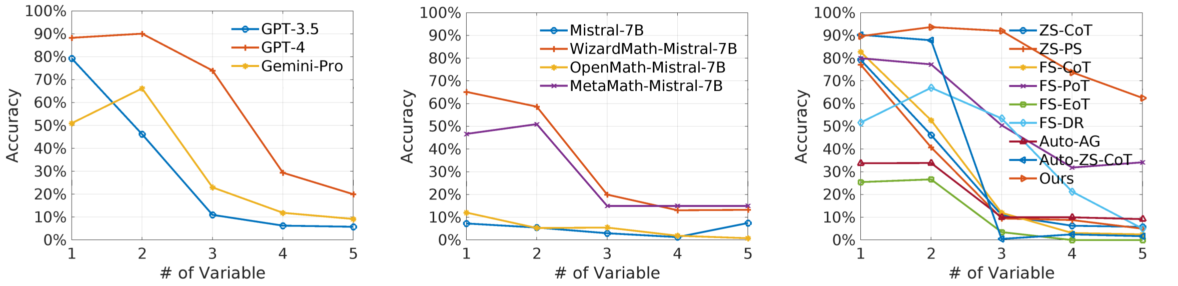
Seed problems. We select ALG514 (Kushman et al., 2014) and DRAW-1K (Upadhyay and Chang, 2017) as the foundational seed problems to expand. These datasets are particularly suitable as they include full solutions with oracle equation sets.

Statistics. With this generation process, we selected a total of 464 problems. Specifically, there are 194 problems with three unknowns, 158 problems with four unknowns, and 112 problems with five unknowns. In addition, since our generated dataset is expanded from the existing dataset, it contains various topics or subjects including moving objects, liquids, interest, distance, and geometry.

4 Benchmarking Existing LLMs and Prompting Methods

4.1 LLMs for solving multi-unknown algebra problems

To evaluate the performance of various LLMs on BeyondX, we consider the Zero-shot-CoT prompting method (details in Section 6.1) and test the performance of both General-Purpose LLMs (GPT-3.5, GPT-4, Gemini-Pro, Mistral-7B) and Mathematically fine-tuned LLMs (WizardMath, OpenMath, MetaMath).



(a) Zero-shot-CoT performance of Close-source model.

(b) Zero-shot-CoT performance of Open-source model.

(c) Different prompting methods performance of GPT-3.5.

Figure 2: Preliminary study of different LLMs and prompting methods on multi-unknown algebra datasets.

In Figure 2a and Figure 2b, the results show a significant performance drop with multiple unknowns on both closed-source and open-source LLMs. For example, GPT-4 achieves near 90% accuracy when solving problems with 1 or 2 unknowns, but the performance drops to 20% when solving problems with 5 unknowns. This highlights that current LLMs with Zero-shot-CoT are not able to solve multi-unknown problems, and this limitation was not recognized in the literature due to lack of datasets. In addition, mathematically fine-tuned LLMs exhibit a significant performance drop when encountering problems with more than two unknowns. This reveals a limitation of current fine-tuning methods, highlighting the need for improved algorithms or training sets.

4.2 Prompting Methods

Figure 2a demonstrates that state-of-the-art LLMs cannot solve multi-unknown problems with Zero-shot-CoT prompting. To investigate whether this issue can be mitigated with better prompting methods, we evaluated nine existing prompting methods using GPT-3.5, categorized into three types:

Zero-shot. Zero-shot-CoT (Kojima et al., 2022) and Plan-and-Solve (Wang et al., 2023) prompting.

Few-shot with manual demonstrations. CoT (Wei et al., 2022), PoT (Gao et al., 2022), EoT (Liu et al., 2023), and Declarative (He-Yueya et al., 2023) prompting.

Few-shot with automatic demonstrations. Analogical (Yasunaga et al., 2023), Auto-Zero-shot-CoT.

In Figure 2c, we observe zero-shot and few-shot CoT prompting methods seem inadequate when solving multi-unknown problems. We find that while CoT correctly sets up the equations, it fails to accurately solve the system of equations. Additionally, even though some prompting methods like PoT, EoT, and Declarative use external tools as a calculator and equation solver, they manually design their demonstration for simpler problems and fail to generalize to more complex multiple unknown scenarios. Although some methods construct demonstrations automatically from the problem context (Analogical, Auto-Zero-shot-CoT), they still suffer from poor performance. Since LLMs themselves do not have enough capability to solve multi-unknown problems, the generated demonstrations are often of low quality. This raises concerns about prompt engineering requiring "human-in-the-loop" solutions with domain knowledge integrated through instructions.

Therefore, in the next section, we will go through a detailed formulation of Formulate-and-Solve prompting. Our method can significantly bridge the gap as shown in Figure 2c.

5 Automatic Solver of Algebra Problems

To investigate whether the observed performance drop is primarily due to inadequate prompting strategies or the limitation of LLMs. we develop **Formulate-and-Solve**, an automated prompting method designed for LLMs to solve math problems with an arbitrary number of unknowns. We also show that our method performs competitively to state-of-the-art algorithms even for non-algebra problems in Appendix A.1.

A major challenge in applying the prompting method to multi-unknown problems is the scarcity of hand-crafted demonstrations. Traditional examples with a single unknown do not scale well to more complex, multi-unknown scenarios, necessitating automated demo generation. Furthermore, while LLMs can be guided by prompts to solve these systems of equations, they often require external tools due to their limited ability to independently solve and explicitly formulate these problems into a system of equations.

To overcome these limitations, we propose Formulate-and-Solve, a framework that incorporates a set of principles to instruct LLMs in generating demonstrations automatically. This framework empowers LLMs to translate problems into equations and subsequently utilize external tools to solve them. The overall pipeline is illustrated in Figure 3 and we include the actual prompts used in each step in Appendix 17.

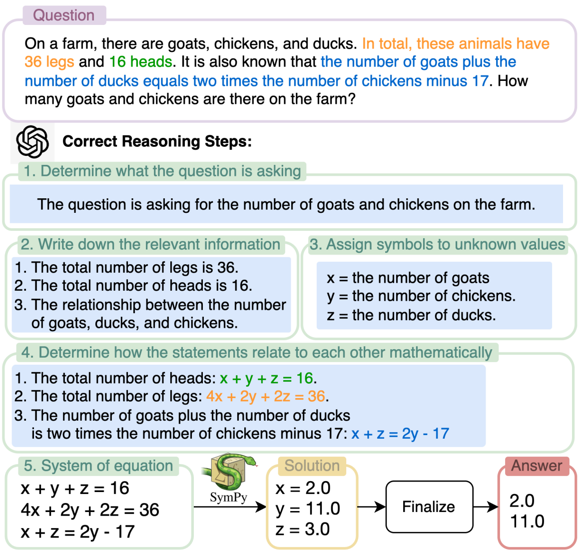


Figure 3: The overview of Formulate-and-Solve.

Automatic generated demonstrations. Conventional prompting methods require creating and evaluating human-written examples to guide LLMs in solving algebra problems, which is time-consuming. Our approach leverages intuitive human-solving steps as instructions. Based on these instructions, the LLM iteratively generates its solution demonstrations (approximately five). To find the most effective demonstrations, we generate ten sets of demonstrations and assess their accuracy on twenty problems. The set with the highest accuracy is chosen as the best.

Solving strategy. Our proposed method leverages the strengths of both LLMs and symbolic solvers. We cooperate with the human-solving steps in the instruction to convert algebra problems into the corresponding systems of equations. Recognizing the limitations of LLMs for complex systems, an external symbolic solver (e.g., SymPy) is employed to solve the system of equations. In the cases of unsolvability or errors that occur in solving the system of equations, the finalization module relies solely on the original prompt and response. This strategy ensures adaptability in some scenarios where the response is not formatted. Furthermore, by incorporating the historical prompt and response within the finalization module, the approach facilitates the continuation or refinement of the solution response.

6 Experimental Results

6.1 Experimental Setting

Dataset. Our experiments are conducted on five algebra problem sets, including existing widely-used ones (ALG514, DRAW-1K, AsDiv, HMWP) and the proposed BeyondX benchmark. AsDiv consists of a wide range of math problems and we only take the algebra problem subset. Also, since HWMP is a Chinese dataset, we use GPT-4 to translate the problem into English. We find that while most translation results effectively convey the intended meaning. The five datasets differ in size and complexity, as shown in Table 1. We also report the average number of unknowns in each dataset. Note that we split the proposed dataset into three subsets, correspond to problems with 3 (BeyondX_3), 4 (BeyondX_4), and 5 (BeyondX_5) unknowns, while the problems in all other datasets have ≤ 2 unknowns.

Dataset	Avg #Unknowns	Avg #Words	Size
ALG514	1.8	41.4	514
DRAW-1K	1.7	35.8	200
AsDiv	1.6	27.9	154
HMWP	1.3	77.3	548
BeyondX_3	3	65.1	194
BeyondX_4	4	94.5	158
BeyondX_5	5	118.8	112

Table 1: Statistics of existing algebra evaluation dataset.

Models. For experiments in this section, we utilize GPT-3.5, GPT-4, and Gemini-Pro as representatives of general-purpose LLMs. We also opt to exclude open-sourced LLMs, as they typically struggle with multi-unknown problems due to their limited capacity to process and follow long prompts. The versions of these models and system prompts we used for the experiments are listed in Appendix D.1.

Baselines. We compare Formulate-and-Solve with three types of prompting baselines: (1) *Zero-shot.* We include Zero-shot-CoT (Kojima et al., 2022) and Plan-and-Solve (PS) (Wang et al., 2023) prompting. The former appends "Let's think step by step" to the prompt without any demo. The latter appends "Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan and solve the problem step by step" to the prompt without any demo. (2) *Few-shot with manual demonstrations.* CoT (Wei et al.,

Setting	Zero-shot		Few-shot (Manual)				Few-shot (Automatic)		
Method	CoT	PS	CoT	PoT†	EoT†	DR†	AG	AZ	Ours†
MU_1	79.3%	77.2%	82.8%	80.0%	25.5%	51.7%	33.8%	90.3%	**89.7%**
MU_2	46.2%	40.8%	52.7%	77.3%	26.7%	67.0%	33.9%	87.9%	**93.7%**
BeyondX_3	11.3%	9.8%	12.4%	52.1%	3.6%	55.2%	10.3%	0.5%	**93.8%**
BeyondX_4	6.3%	8.9%	3.2%	32.3%	0.0%	21.5%	10.1%	2.5%	**84.9%**
BeyondX_5	6.2%	5.4%	2.7%	36.6%	0.0%	5.4%	9.8%	1.8%	**71.4%**
Average	29.9%	28.4%	30.7%	55.7%	11.2%	40.1%	19.6%	36.6%	**86.7%**

Table 2: Experiment results across various unknowns using GPT-3.5. † means the method uses external tools.

Setting	Zero-shot		Few-shot (Manual)				Few-shot (Automatic)		
Method	CoT	PS	CoT	PoT†	EoT†	DR†	AG	AZ	Ours†
				Single					
ALG514	91.2%	85.7%	91.2%	93.4%	31.9%	59.3%	38.5%	**93.4%**	92.3%
DRAW-1K	59.3%	63.0%	68.5%	57.4%	14.8%	38.9%	25.9%	85.2%	**85.2%**
AsDiv	68.3%	71.4%	71.4%	75.0%	26.8%	53.5%	54.0%	76.1%	**81.7%**
HMWP	21.5%	14.6%	22.0%	25.2%	4.2%	10.3%	21.0%	31.0%	**37.7%**
Average	60.1%	58.7%	63.3%	62.8%	19.4%	40.5%	34.8%	71.4%	**74.2%**
				Double					
ALG514	47.0%	40.7%	53.4%	81.8%	29.6%	68.8%	34.5%	90.5%	**96.5%**
DRAW-1K	43.8%	41.1%	50.7%	64.4%	18.5%	61.6%	32.2%	80.1%	**85.6%**
AsDiv	55.0%	49.5%	37.4%	11.0%	15.7%	59.0%	31.9%	79.5%	**80.7%**
HMWP	16.4%	9.9%	18.7%	39.8%	9.9%	26.3%	28.1%	50.3%	**58.5%**
Average	40.6%	35.3%	40.0%	49.2%	18.4%	53.9%	31.7%	75.1%	**80.3%**

Table 3: Experiment results across various algebra problem datasets which include single and double unknown using GPT-3.5. † means the method uses external tools.

2022) creates eight hand-crafted natural language examples as demonstrations. PoT (Gao et al., 2022) creates eight hand-crafted Python code examples as demonstrations and uses programming tools to get the final answer. EoT (Liu et al., 2023) creates eight hand-crafted equation examples as demonstrations and uses symbolic solvers to obtain the final answer. Declarative (DR) (He-Yueya et al., 2023) creates three hand-crafted examples with principles as demonstrations and uses symbolic solvers to obtain the final answer. (3) *Few-shot with automatic demonstrations.* (Yasunaga et al., 2023) proposed Analogical prompting (AG), designed to automatically guide LLMs to self-generate relevant examples as demonstrations before proceeding to solve the problem. We come up with another naive method: selecting examples from the dataset and employing Zero-shot-CoT (Kojima et al., 2022) to generate examples as demonstrations. We refer to this method as Auto-Zero-shot-CoT (AZ).

6.2 Main Results

To investigate which reasoning methods and models better solve multi-unknown problems, we summarize the performance of different prompting methods using GPT-3.5 in Table 2 and Tables 3. We also evaluate and compare various prompting methods with GPT-4 and Gemini-Pro in Appendix A.3.

Comparison with various prompting baselines. Table 2 and Figure 2c present the accuracy comparison of our method with existing approaches on datasets containing 1 to 5 unknowns. We combine ALG514 and DRAW-1K into MU_1 and MU_2 for problems with 1 and 2 unknowns, respectively, and use BeyondX for problems with 3, 4, or 5 unknowns. Figure 2c illustrates that only our method maintains reasonable accuracy when the number of unknowns exceeds two, while other methods experience a significant performance decline. For problems with five unknowns, our method achieves a 66.7% accuracy, while the best alternative achieves only 34.2%.

This improvement is attributed to our instructional approach and automatic demonstrations, which effectively address general algebra problems. Furthermore, datasets with multiple unknowns typically involve longer questions and necessitate the construction of more equations for a solution. We find that using lengthy natural language reasoning steps can easily introduce operational and calculation errors. In contrast, our method guides LLMs to generate a system of equations as an intermediate reasoning step, making it less prone to mistakes during equation formulation. As a result, our method maintains high accuracy when using a symbolic solver to solve the equations. We also observe that leveraging external tools, such as programming or symbolic solvers, to tackle algebra problems generally yields better performance than directly obtaining the final answer from the model in the

Instruction	Demos	Solver	Equation	U1	U2	U3	U4	U5
✗	✗	✓	✓	77.8%	74.0%	60.0%	33.8%	18.3%
✗	✓	✓	✓	79.6%	77.4%	79.0%	56.3%	48.3%
✓	✗	✓	✓	64.1%	81.2%	82.0%	68.3%	60.0%
✓	✓	✗	✓	59.3%	73.3%	43.0%	16.8%	15.0%
✓	✓	✓	✓	**85.2%**	**87.6%**	**90.0%**	**81.3%**	**66.7%**

Table 4: Ablation experiment results across various number of unknowns using GPT-3.5. Our method achieves the highest accuracy among all.

presence of multiple unknowns.

Next, we verify whether Formulate-and-Solve maintains its effectiveness for problems with one or two unknowns. For this, we compare it with baselines on commonly used algebra datasets containing one or two unknowns, with the results presented in Table 3. The results demonstrate that Formulate-and-Solve again achieves the best performance. Compared with other automatic few-shot methods such as AZ, the performance gap is considerably smaller (5.2% on average) for one or two unknowns than for multiple unknowns (48.4% on average). This is likely because it is easier to generate a high-quality demo for problems with one or two unknowns. Also, we cannot see a big difference between the zero-shot and few-shot CoT in this experiment since the manual few-shot demonstrations that are commonly used in previous work are beneficial for solving arithmetic problems, not algebra problems.

7 Discussion and Analysis

Ablation study. We analyze the significance of each component within in Formulate-and-Solve through an ablation study. We assess five variations: (1) Use only a system of equations as a rationale for reasoning. (2) Remove the instruction before demonstrations. (3) Remove demonstrations after the instruction. (4) Use an LLM instead of a symbolic solver to solve a system of equations. (5) Formulate-and-Solve. We randomly select 60 problems for each unknown from ALG514, DRAW-1K, and BeyondX to evaluate each variation. The results are provided in Table 4.

We observe that performance decreases significantly when instruction or demonstration is removed, highlighting its role in guiding the LLM. Interestingly, instruction has a greater impact than demonstration. Replacing the symbolic solver with an LLM also leads to a large decrease in accuracy. These findings confirm that all elements in Formulate-and-Solve contribute significantly to solving multi-unknown problems.

Error Reason	Percentage (%)
(E1) Fewer or More Equations	17.4% (12/69)
(E2) Incorrect Equation	44.9% (31/69)
(E3) Incorrect Extraction Form	37.7% (26/69)

Table 5: Statistics of error analysis under GPT-3.5.

Error analysis. We delve deeper into the primary challenges that LLMs encounter when solving multi-unknown problems using Formulate-and-Solve. To gain a quantitative understanding of model failures, we conduct an error analysis in Formulate-and-Solve with GPT-3.5 on BeyondX. We collect all instances where predictions were incorrect and annotate the main reasons for these mispredictions. The error types include: (E1) generating too few or too many equations, (E2) producing the correct number of equations but with incorrect content, and (E3) generating responses in the wrong format, preventing the extraction of the equation system.

As illustrated in Table 5, the most common error is E2 (incorrect equation). This indicates that current LLMs equipped with prompting methods still struggle to accurately formulate multi-unknown equations in some cases. Besides, 37.7% of the errors occur due to the wrong format of the response, and 17.4% of the errors arise when LLMs fail to align relevant information correctly with the equations, resulting in either too few or too many equations. The detailed qualitative analysis of the error examples is in Appendix D.3.

Dataset	Out-of-Domain	In-Domain
MU_1	82.8%	82.4%
MU_2	52.7%	48.7%
BeyondX_3	12.0%	10.4%
BeyondX_4	3.1%	6.4%
BeyondX_5	2.5%	1.9%

Table 6: In- and Out-Domain demonstrations analysis under GPT-3.5 via Few-shot-CoT.

Effectiveness of in-domain and out-of-domain demonstrations. To investigate the impact of in-domain demonstrations, we manually crafted five demonstrations tailored to our multi-unknown problems. In Table 6, we compare the performance of Few-shot-CoT prompting using these in-domain and out-of-domain demonstrations. Although in-domain demonstrations can be beneficial, our results observe that factors like the quality and structure of demonstrations are also crucial.

8 Conclusion

We introduce BeyondX, the first benchmark for evaluating LLMs on multi-unknown problems. Our analysis reveals a significant performance drop in LLMs and existing prompting methods when faced with such problems. To address this, we propose Formulate-and-Solve, a novel prompting method that leverages instruction, automatic demonstrations and a system of equations. Experiments demonstrate the effectiveness of Formulate-and-Solve in tackling multi-unknown problems.

Limitations

Scope of Benchmark.

Although our automatic generation method can decrease the labor-intensive data collection process, our method still needs to be expanded from high-quality problems with low unknowns. Besides, we figure that some types or topics of the problems cannot be extended to multiple unknown problems. And, our benchmark is limited to English questions and data. We look forward to future benchmarks on a broader domain or modality and other languages.

Models and Reasoning Methods.

Although we experiment with many representative models and reasoning methods in this paper, we acknowledge that this does not cover all models and frameworks. Besides, we acknowledge that our approach falls short on more straightforward arithmetic datasets since our method is more suitable for algebra datasets. Further research is required to explore new problem-solving methods for general math reasoning tasks, including different modalities.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.

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

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2022. Compositional semantic parsing with large language models. In *The Eleventh International Conference on Learning Representations*.

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

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

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

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

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*, 3:585–597.

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

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland. Association for Computational Linguistics.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Chao-Chun Liang, Kuang-Yi Hsu, Chien-Tsung Huang, Chung-Min Li, Shen-Yu Miao, and Keh-Yih Su. 2016. A tag-based english math word problem solver with understanding, reasoning and explanation. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Demonstrations*, pages 67–71.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Haoxiong Liu and Andrew Chi-Chih Yao. 2024. Augmenting math word problems via iterative question composing. *arXiv preprint arXiv:2401.09003*.

Tengxiao Liu, Qipeng Guo, Yuqing Yang, Xiangkun Hu, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2023. Plan, verify and switch: Integrated reasoning with diverse x-of-thoughts. *arXiv preprint arXiv:2310.14628*.

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

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online. Association for Computational Linguistics.

Kumaresh Nandhini and Sadhu Ramakrishnan Balasundaram. 2011. Math word question generation for training the students with learning difficulties. In *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*, pages 206–211.

Kole Norberg, Husni Almoubayyed, Stephen E Fancsali, Logan De Ley, Kyle Weldon, April Murphy, and Steven Ritter. 2023. Rewriting math word problems with large language models. In *AIEd23: artificial intelligence in education, empowering education with LLMs workshop*.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. 2021. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. 2023. Openwebmath: An open dataset of high-quality mathematical web text. *arXiv preprint arXiv:2310.06786*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Oleksandr Polozov, Eleanor O'Rourke, Adam M Smith, Luke Zettlemoyer, Sumit Gulwani, and Zoran Popović. 2015. Personalized mathematical word problem generation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2022. Reasoning with language model prompting: A survey. *arXiv preprint arXiv:2212.09597*.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. *Transactions of the Association for Computational Linguistics*, 6:159–172.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv preprint arXiv:2402.10176*.

Shyam Upadhyay and Ming-Wei Chang. 2017. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 494–504, Valencia, Spain. Association for Computational Linguistics.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

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

Zichao Wang, Andrew Lan, and Richard Baraniuk. 2021. Math word problem generation with mathematical consistency and problem context constraints. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5986–5999, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Sandra Williams. 2011. Generating mathematical word problems. In *2011 AAAI Fall symposium series*.

Michihiro Yasunaga, Xinyun Chen, Yujia Li, Panupong Pasupat, Jure Leskovec, Percy Liang, Ed H Chi, and Denny Zhou. 2023. Large language models as analogical reasoners. *arXiv preprint arXiv:2310.01714*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

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

Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. 2024. Mammoth2: Scaling instructions from the web. *arXiv preprint arXiv:2405.03548*.

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

Mingyu Zong and Bhaskar Krishnamachari. 2023. Solving math word problems concerning systems of equations with gpt-3. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 15972–15979.

Algorithm 1 Formulate-and-Solve Reasoning Algorithm

Require: question Q, instruction I, reasoning module R, symbolic solver S, finalize module F

function AUTO DEMO($I, Question, K$)
  while $K \neq 0$ **do**
    $D \leftarrow R(I + D + Question)$
    $K \leftarrow K - 1$
  end while
  return D ▷ D is a demo
end function
$D \leftarrow AutoDemo()$
$p \leftarrow I + D + Q$ ▷ p is a input prompt
$eq \leftarrow R(p)$
if $S(eq)$ **then** ▷ Equation System is solvable
  $ans \leftarrow F(Q + eq + S(eq))$
else
  $ans \leftarrow F(Q + eq)$
end if
return ans ▷ Return the Answer

A Further Experiment

A.1 Generalization to common arithmetic datasets

We analyze the generalizability of the Formulate-and-Solve framework to other common arithmetic datasets, such as GSM8K, SVAMP, AddSub, SingleEq, and MultiArith where some problems can be seen as single unknown problems on GPT-3.5. The details of these arithmetic datasets can be seen in Table 8. From Table 9, we can observe that our method can still perform a comparable performance to other existing custom and manual prompting methods for arithmetic tasks since these datasets are much easier than multiple unknown datasets and allow only minimal room for improvement. Besides, our method is more generalized not only to single unknown problems but also can deal with multiple unknown problems automatically.

A.2 Experiments different mathematical models on BeyondX

We further evaluate seven different existing mathematical models fine-tuned on Mistral-7B on **BeyondX** under Zero-shot-CoT setting. As shown in Figure 4 and Table 10, the results indicate that these open-source LLMs are still struggling with more complex mathematical reasoning tasks in multiple unknown problems. There is still a significant amount of effort during pretraining or supervised fine-tuning to instill enough multiple unknown knowledge and the way of solving multiple unknown system of equations into the models' parameters to close the gap.

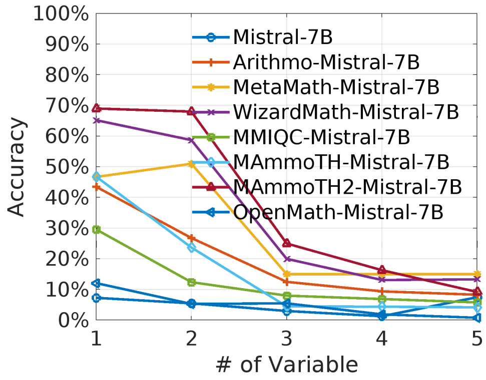


Figure 4: The performance of different existing open-source models.

A.3 Experiments different models on Formulate-and-Solve

We further assess the performance of Formulate-and-Solve across various base models, such as GPT-4 and Gemini-Pro. The results of the general algebra dataset and the multi-unknown dataset are shown in Table 14 and Table 15, and we also illustrate the performance curve of GPT-4 and Gemini-Pro on different numbers of unknown in Figure 5a and Figure 5b. The findings remain as GPT-3.5 and our method outperforms a large gap among other methods. Additionally, we observe that the performance of the Gemini-Pro generally falls between that of GPT-3.5 and GPT-4 across various datasets and prompting methods.

A.4 Experiments different shots of demonstrations on Formulate-and-Solve

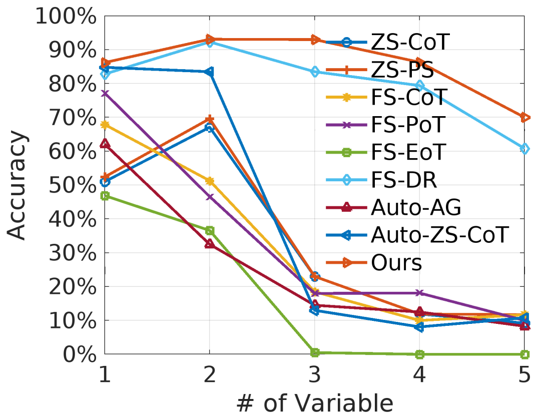
We analyze the effect of varying the number of automatic generated exemplars (K) in our approach on GPT-3.5. Here, we show three variations with K = 3, 5, and 8. In Table 11, we observe that LLM demonstrates consistent performance under single or double unknown in different datasets. When K is bigger, on average, performance improves.

B Full Instruction

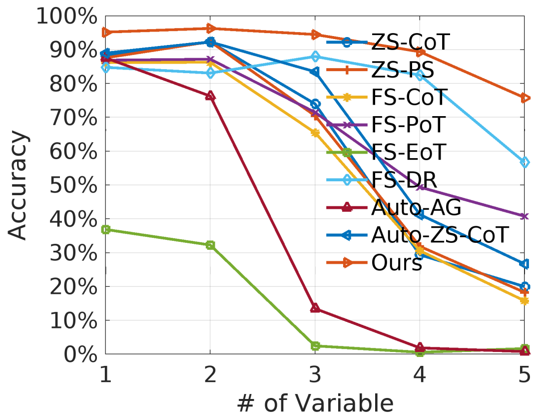
In this section, we show the full instructions in Section 3 and Section 5.

Dataset	Rationale	Size	# of Variables	Source	Domain
SingleEQ	Equation	508	1	Internet	Arithmetic
MAWPS	Equation	3320	1	Internet	Arithmetic
AllArith	Equation	831	1	Internet	Arithmetic
Dolphin18K	Equation	18460	1	Internet	Arithmetic/Algebra
Math23K	Equation	23162	1	Internet	Arithmetic/Algebra
SVAMP	Equation	1000	1	Internet	Arithmetic/Algebra
GSM8K	Natural Language	8792	1	Annotated	Arithmetic/Algebra
AQuA	Natural Language	100000	1	GMAT/GRE	Arithmetic/Algebra
MATHQA	Natural Language	37297	1	GMAT/GRE	Arithmetic/Algebra
ASDiv	Equation	2305	1-2	Internet	Arithmetic/Algebra
ALG514	Equation	514	1-2	Internet	Algebra
DRAW-1K	Equation	1000	1-2	Internet	Algebra
HMWP	Equation	5470	1-2	Internet	Algebra
BeyondX	**Equation**	**480**	**1-5**	**LLMs Generated**	**Algebra**

Table 7: List of existing math dataset.



(a) Different prompting methods performance of Gemini-Pro.



(b) Different prompting methods performance of GPT-4.

Figure 5: The performance on different numbers of unknown.

Dataset	Domain	Avg #Unknowns	Evaluation Size
SingleEq	Arithmetic	1	508
MultiArith	Arithmetic	1	600
AddSub	Arithmetic	1	395
SVAMP	Arithmetic/Algebra	1	1000
GSM8K	Arithmetic/Algebra	1	1319
ALG514	Algebra	1.8	514
DRAW-1K	Algebra	1.7	200
BeyondX	**Algebra**	**3.8**	**464**

Table 8: Statistics of existing math evaluation dataset.

B.1 Automatic generation of multi-unknown algebra problems

We can see the full instructions in Table 16.

B.2 Automatic solver of multi-unknown algebra problems

We can see the full instructions in Table 17.

C Detail Studies of Automatic Generation of Multi-Unknown Algebra Problems

C.1 Construction steps

Starting from the 2 unknown problems in our seed dataset ALG514 and DRAW-1K, we use Zero-shot prompting with GPT-4 (`gpt-4-0613`) to generate an initial demonstration using the instruction in Table 16, which is then manually refined. The LLM then iteratively creates additional demonstrations (approximately five) based on the problem, the system of equations, and the existing demonstrations. Combining this information, the LLM generates a new problem with N+1 unknowns and its corresponding system of equations. Finally, we use GPT-4 to solve these newly generated problems, discarding low-quality ones (where GPT-4 provides incorrect answers) for the next round.

C.2 Quality validation

We recruit 12 raters to validate whether the generated problems are reasonable and whether they are consistent with the generated system of equations. We show the ratio of problems that are marked as "unreasonable" by raters in Table 12. To understand why LLMs struggle with our instruction, we analyzed unreasonable problems in Table 20. Our

Setting	Zero-shot		Few-shot (Manual)				Few-shot (Automatic)		
Method	CoT	PS	CoT	PoT†	EoT†	DR†	AG	CoT	Ours†
MultiArith	94.3%	95.3%	**98.7%**	<u>98.2%</u>	50.2%	90.5%	73.8%	95.7%	97.7%
GSM8K	<u>77.9%</u>	75.1%	**79.5%**	75.7%	28.2%	59.4%	52.9%	77.7%	71.4%
AddSub	91.9%	89.9%	**94.9%**	92.4%	55.4%	89.6%	64.8%	<u>94.9%</u>	91.5%
SingleEq	95.7%	97.0%	**98.4%**	<u>97.6%</u>	53.0%	92.3%	67.5%	<u>97.6%</u>	96.3%
SVAMP	<u>82.7%</u>	82.1%	80.8%	**84.8%**	44.9%	76.9%	59.5%	82.1%	81.6%
Average	88.5%	87.6%	**90.5%**	<u>89.7%</u>	46.3%	81.7%	63.7%	89.6%	87.7%

Table 9: We compare the results across various arithmetic problem datasets using GPT-3.5. † means the method uses external tools.

Model name	Dataset	Tool-Integrated	MU_1	MU_2	BeyondX_3	BeyondX_4	BeyondX_5
Mistral-7B (Base)	-	-	7.3%	5.5%	3.1%	1.3%	8.0%
Arithmo-Mistral-7B	MetaMathQA &MathInstruct	✗	43.5%	26.8%	12.9%	9.5%	8.9%
MetaMath-Mistral-7B	MetaMathQA	✗	46.7%	51.0%	15.5%	<u>15.2%</u>	**16.1%**
WizardMath-Mistral-7B	Not released	✗	<u>65.2%</u>	<u>58.7%</u>	<u>20.6%</u>	13.3%	<u>14.3%</u>
MMIQC-Mistral-7B	MMIQC	✗	29.6%	12.4%	8.2%	7.0%	6.2%
MAmmoTH-Mistral-7B	MathInstruct	✓	46.8%	23.7%	4.6%	4.4%	4.5%
MAmmoTH2-Mistral-7B	WebInstruct	✓	**69.0%**	**68.0%**	**25.8%**	**16.5%**	9.8%
OpenMath-Mistral-7B	OpenMathInstruct-1	✓	12.1%	5.3%	5.7%	1.9%	0.9%

Table 10: Experiment results of open-source math models that are fine-tune on Mistral-7B base model across various algebra problem datasets under Zero-shot-CoT setting.

K-Shot	3-shot	5-shot	8-shot
		Single	
ALG514	92.3%	<u>92.3%</u>	**95.6%**
DRAW-1K	82.6%	<u>85.2%</u>	**90.7%**
AsDiv	79.7%	<u>81.7%</u>	**85.5%**
HMWP	34.5%	<u>37.7%</u>	**38.0%**
Average	72.3%	<u>74.2%</u>	**77.5%**
		Double	
ALG514	96.0%	<u>96.5%</u>	**97.1%**
DRAW-1K	84.9%	<u>85.6%</u>	**85.6%**
AsDiv	**82.7%**	80.7%	<u>81.7%</u>
HMWP	57.9%	<u>58.5%</u>	**60.9%**
Average	<u>80.4%</u>	80.3%	**81.3%**

Table 11: Performance comparison of our method across different shots using GPT-3.5.

Source Dataset	Avg #unknowns	Size	Avg #words	Unreasonable Problem Rate
ALG514	3	97	68.8	25.4% (33/130)
	4	78	96.6	20.4% (20/98)
	5	55	128.4	11.3% (7/62)
DRAW-1K	3	97	61.5	30.2% (42/139)
	4	80	92.4	25.2% (27/107)
	5	57	109.2	14.9% (10/67)

Table 12: Statistics of Proposed dataset. Unreasonable Problem Rate means #Unreasonable Problem/ #Total Human Seen Problem

findings reveal several limitations. First, LLMs cannot directly derive complex constant meanings requiring decomposition (Case 1). Second, unclear instructions lead to repetitive equations (Case 2). Third, introducing new variables might not be effective for all problems (e.g., river rate and distance, Case 3). Finally, LLMs may generate inconsistent numerical values within problems (Case 4).

C.3 Full examples

In Figure 6, we show the full examples of our proposed generation method under each unknown.

D Detail Studies of Automatic Solver of Algebra Problems

D.1 Model hyperparameters

The hyperparameters for the experiments for studying Formulate-and-Solve and other prompting methods are set to their default values to ensure consistency in our experiment. Table 13 details the specific generation parameters for the various LLMs we evaluate.

D.2 Overall pipeline

We describe the overall pipeline of Formulate-and-Solve in Algorithm 1.

D.3 Qualitative analysis of error cases

We show every type of error case that GPT-3.5 cannot answer correctly in Table 21. From E1, the system of equations is missing an equation about the relation "Total number of cars: 20". From E2, the first equation is wrong since the relation is "Total sum of the average miles per gallon obtained by the three cars is 75", which means "a + b + c = 75". From E3, since the response format is different from the demonstration, we cannot extract the system of equations from the response.

Model	Version	Generation Setup
	Close-source	
GPT-3.5	gpt-3.5-turbo-1106	temperature = 0, max tokens = 2048
GPT-4	gpt-4-1106-preview	temperature = 0, max tokens = 2048
Gemini-Pro	gemini-1.0-pro	temperature = 0, max tokens = 2048
	Open-source	
Mistral-7B	mistralai/Mistral-7B-v0.1	do_sample=True, temperature=0.001, max tokens = 2048
Arithmo-Mistral-7B	akjindal53244/Arithmo-Mistral-7B	do_sample=True, temperature=0.001, max tokens = 2048
MetaMath-Mistral-7B	meta-math/MetaMath-Mistral-7B	do_sample=True, temperature=0.001, max tokens = 2048
WizardMath-Mistral-7B	WizardLM/WizardMath-7B-V1.1	do_sample=True, temperature=0.001, max tokens = 2048
MMIQC-Mistral-7B	Vivacem/Mistral-7B-MMIQC	do_sample=True, temperature=0.001, max tokens = 2048
MAmmoTH-Mistral-7B	TIGER-Lab/MAmmoTH-7B-Mistral	do_sample=True, temperature=0.001, max tokens = 2048
MAmmoTH2-Mistral-7B	TIGER-Lab/MAmmoTH2-7B	do_sample=True, temperature=0.001, max tokens = 2048
OpenMath-Mistral-7B	nvidia/OpenMath-Mistral-7B-v0.1-hf	do_sample=True, temperature=0.001, max tokens = 2048

Table 13: Model names, versions, and generating setups for various LLMs.

Model	Setting	Zero-shot		Few-shot (Manual)				Few-shot (Automatic)		
		CoT	PS	CoT	PoT†	EoT†	DR†	AG	AZ	Ours†
Gemini	MU_1	51.0%	52.4%	67.9%	77.2%	46.9%	82.8%	62.1%	<u>84.8%</u>	**86.2%**
	MU_2	67.1%	69.6%	51.3%	46.6%	36.6%	<u>92.3%</u>	32.5%	83.5%	**93.1%**
	BeyondX_3	23.7%	23.7%	19.1%	18.6%	0.5%	<u>86.1%</u>	14.9%	13.4%	**95.9%**
	BeyondX_4	12.0%	12.0%	10.1%	18.4%	0.0%	<u>80.4%</u>	12.7%	8.2%	**87.3%**
	BeyondX_5	9.8%	12.5%	12.5%	10.7%	0.0%	<u>65.2%</u>	8.9%	11.6%	**75.0%**
	Average	32.7%	34.1%	32.2%	34.3%	16.8%	<u>81.3%</u>	26.2%	40.3%	**87.5%**
GPT-4	MU_1	88.3%	87.6%	86.2%	86.9%	36.9%	84.8%	87.6%	<u>89.0%</u>	**95.2%**
	MU_2	92.3%	<u>92.4%</u>	86.3%	87.2%	32.3%	83.1%	76.3%	92.3%	**96.3%**
	BeyondX_3	76.3%	72.7%	67.5%	73.7%	2.6%	<u>90.7%</u>	13.9%	86.1%	**97.4%**
	BeyondX_4	29.7%	32.3%	31.0%	50.0%	0.6%	<u>83.5%</u>	1.9%	41.8%	**90.5%**
	BeyondX_5	21.4%	19.6%	17.0%	43.8%	1.8%	<u>60.7%</u>	0.9%	28.6%	**81.3%**
	Average	61.6%	60.9%	57.6%	68.3%	14.8%	<u>80.6%</u>	36.1%	67.5%	**92.1%**

Table 14: Experiment results across various unknowns using Gemini-Pro and GPT-4. † means whether the method uses external tools.

D.4 Full examples

In Table 22 and Table 23, we show the input and output examples for each unknown using Formulate-and-Solve in GPT-3.5.

Model	Setting	Zero-shot		Few-shot (Manual)				Few-shot (Automatic)		
		CoT	PS	CoT	PoT†	EoT†	DR†	AG	AZ	Ours†
					Single					
	ALG514	55.0%	55.0%	82.4%	89.0%	52.8%	86.8%	70.3%	89.0%	**94.5%**
	DRAW-1K	44.4%	48.2%	43.3%	57.4%	37.0%	75.9%	48.2%	**77.8%**	72.2%
Gemini	AsDiv	47.9%	49.3%	57.8%	75.0%	60.6%	74.7%	49.3%	74.7%	**77.5%**
	HMWP	18.6%	17.0%	22.3%	15.1%	23.6%	23.1%	26.0%	30.8%	**46.6%**
	Average	41.5%	42.3%	51.4%	59.1%	43.5%	65.1%	48.4%	68.1%	**72.7%**
	ALG514	92.3%	93.4%	92.3%	90.1%	39.9%	86.8%	93.4%	93.4%	**98.9%**
	DRAW-1K	81.5%	77.8%	75.9%	81.5%	31.9%	81.5%	77.8%	81.5%	**88.9%**
GPT-4	AsDiv	88.9%	88.9%	88.9%	62.5%	11.3%	81.7%	84.1%	83.1%	**90.1%**
	HMWP	54.9%	52.8%	53.1%	22.0%	0.3%	31.3%	32.6%	**55.4%**	54.9%
	Average	79.4%	78.2%	77.5%	64.0%	20.8%	70.3%	72.0%	78.4%	**83.2%**
					Double					
	ALG514	69.3%	69.3%	53.9%	47.8%	39.7%	94.8%	33.3%	86.8%	**95.3%**
	DRAW-1K	61.0%	70.6%	43.8%	43.2%	27.4%	84.9%	30.1%	74.0%	**87.0%**
Gemini	AsDiv	77.1%	74.7%	63.9%	7.5%	7.2%	85.5%	49.4%	74.7%	**84.3%**
	HMWP	25.2%	32.2%	32.8%	26.9%	2.9%	56.7%	25.7%	43.9%	**61.7%**
	Average	58.1%	61.7%	48.6%	31.3%	19.3%	80.5%	34.7%	69.8%	**82.1%**
	ALG514	97.2%	96.9%	91.5%	89.8%	32.4%	84.4%	80.1%	94.3%	**98.8%**
	DRAW-1K	78.1%	79.5%	71.2%	79.5%	32.1%	79.5%	65.1%	86.3%	**89.0%**
GPT-4	AsDiv	86.8%	85.7%	83.5%	11.6%	2.4%	79.5%	39.6%	**91.6%**	90.4%
	HMWP	64.3%	64.9%	57.3%	56.3%	1.2%	57.9%	38.6%	71.4%	**75.4%**
	Average	81.6%	81.8%	75.9%	59.3%	17.0%	75.3%	55.8%	85.9%	**88.4%**

Table 15: Experiment results across various algebra problem datasets which include single and double unknown using Gemini-Pro and GPT-4. † means whether the method uses external tools.

Instruction for Automatic Generation of Multiple Unknown Algebra Problems
1. Write down each variable meaning in a system of equations by understanding the relation between a system of equations and a math word problem. 2. Introduce a new variable based on the existing variable meaning. Then, assign a value to the new variable based on the solution of the existing system of equations. 3. Add new terms and new equations to the existing system of equations to generate a solvable three-unknown system of equations. 4. Introduce new statements that are related to the existing math word problem or modify the existing statement to rewrite the problem into a three-unknown math word problem based on the new system of equations and the same original scenario. 5. Formulate and rephrase the statements and scenario into a coherent and reasonable math word problem.

Table 16: Actual instruction of algebra problem generation.

Instruction for Automatic Solver of Algebra Problems
1. Determine what the question is asking. 2. Write down the relevant information in simple statements. 3. Assign symbols (must be an alphabetic character e.g., x, y, z etc.) to unknown values that must be found. 4. Determine how the statements relate to each other mathematically. 5. Give the equations only here, with each equation on a new line.

Table 17: Actual instruction of algebra problem solver.

MWP:	Suppose you invested 10,000 dollars, part at 6 % annual interest and the rest at 9 % annual interest. If you received 684 dollars in interest after one year, how much did you invest at 6 % annual interest and at 9 % annual interest?
System of Equations	$6.0 * 0.01 * x + 9.0 * 0.01 * y = 684.0$ $x + y = 10000.0$ $(x = 7200.0 ; y = 2800.0)$
Variable Meaning	x = invested at 6% annual interest ; y = invested at 9% annual interest

(a) Two-unknown (source problem)

MWP:	Suppose you invested 12,000 dollars in three different ways: part at 6 % annual interest, part at 9 % annual interest, and the rest at 4 % annual interest. After one year, you received 764 dollars in interest. Also, the amount invested at 6% annual interest plus the amount invested at 9% annual interest equals five times the amount invested at 4% annual interest. How much did you invest at each rate?
System of Equations	$6.0 * 0.01 * x + 9.0 * 0.01 * y + 4.0 * 0.01 * z = 764.0$ $x + y + z = 12000.0$ $x + y = 5.0 * z$ $(x = 7200.0 ; y = 2800.0 ; z = 2000.0)$
Variable Meaning	x = invested at 6% annual interest ; y = invested at 9% annual interest ; z = invested at 4% annual interest

(b) Three-unknown

MWP:	Suppose you invested 14,000 dollars in four different ways: part at 6 % annual interest, part at 9 % annual interest, part at 4 % annual interest, and the rest at 3 % annual interest. After one year, you received 824 dollars in interest. Also, the amount invested at 6% annual interest plus the amount invested at 9% annual interest equals five times the amount invested at 4% annual interest. The amount invested at 6% annual interest plus the amount invested at 9% annual interest plus the amount invested at 4% annual interest equals six times the amount invested at 3% annual interest. How much did you invest at each rate?
System of Equations	$6.0 * 0.01 * x + 9.0 * 0.01 * y + 4.0 * 0.01 * z + 3.0 * 0.01 * w = 824.0$ $x + y + z + w = 14000.0$ $x + y = 5.0 * z$ $x + y + z = 6.0 * w$ $(x = 7200.0 ; y = 2800.0 ; z = 2000.0 ; w = 2000.0)$
Variable Meaning	x = invested at 6% annual interest ; y = invested at 9% annual interest ; z = invested at 4% annual interest ; w = amount invested at 3% annual interest

(c) Four-unknown

MWP:	Suppose you invested 16,000 dollars in five different ways: part at 6 % annual interest, part at 9 % annual interest, part at 4 % annual interest, part at 3 % annual interest, and the rest at 2 % annual interest. After one year, you received 864 dollars in interest. Also, the amount invested at 6% annual interest plus the amount invested at 9% annual interest equals five times the amount invested at 4% annual interest. The amount invested at 6% annual interest plus the amount invested at 9% annual interest plus the amount invested at 4% annual interest equals six times the amount invested at 3% annual interest. The amount invested at 6% annual interest plus the amount invested at 9% annual interest plus the amount invested at 4% annual interest plus the amount invested at 3% annual interest equals seven times the amount invested at 2% annual interest. How much did you invest at each rate?
System of Equations	$6.0 * 0.01 * x + 9.0 * 0.01 * y + 4.0 * 0.01 * z = 864.0$ $x + y + z = 12000.0$ $x + y = 5.0 * z$ $x + y + z = 6.0 * w$ $x + y + z + w = 7.0 * v$ $(x = 7200.0 ; y = 2800.0 ; z = 2000.0 ; w = 2000.0 ; v = 2000.0)$
Variable Meaning	x = invested at 6% annual interest ; y = invested at 9% annual interest ; z = invested at 4% annual interest ; w = amount invested at 3% annual interest ; v = amount invested at 2% annual interest

(d) Five-unknown

Figure 6: Example problems with different numbers of unknown generated from the same source problem.

Prompt:
Generate a three-unknown math word problem and its corresponding system of equations.
Example 1: Word Problem: Three friends, Alice, Bob, and Carol, decide to pool their money to buy a gift for a friend. Alice contributes twice as much as Bob, while Carol contributes $10 more than Bob. The total amount they collect is $110. How much did each person contribute? System of Equations: Let A represent the amount Alice contributes,B represent the amount Bob contributes, and C represent the amount Carol contributes. Based on the information given: 1. A = 2B (Alice contributes twice as much as Bob) 2. C = B + 10 (Carol contributes $10 more than Bob) 3. A + B + C = 110 (The total amount collected is $110)
Example 2: Word Problem: During a school fundraiser, three students, Emily, Daniel, and Fiona, sold raffle tickets. Emily sold 5 more tickets than Daniel. Fiona sold twice as many tickets as Emily. Together, they sold a total of 155 tickets. How many tickets did each student sell? System of Equations: Let E represent the number of tickets Emily sold, D represent the number of tickets Daniel sold, and F represent the number of tickets Fiona sold. Based on the information given: 1. E = D + 5 (Emily sold 5 more tickets than Daniel) 2. F = 2E (Fiona sold twice as many tickets as Emily) 3. E + D + F = 155 (Together, they sold a total of 155 tickets)
Example 3: Word Problem: In a small bakery, three types of pastries are made daily: croissants, muffins, and scones. The number of croissants made is twice the number of muffins. The bakery made 30 more scones than muffins. At the end of the day, they counted a total of 210 pastries. How many of each type of pastry were made? System of Equations: Let C represent the number of croissants, M represent the number of muffins, and S represent the number of scones. Based on the information given: 1. C = 2M (The number of croissants is twice the number of muffins) 2. S = M + 30 (There are 30 more scones than muffins) 3. C + M + S = 210 (The total number of pastries is 210)

Table 18: Examples of multiple unknown algebra problem and corresponding system of equations using simple prompt on GPT-4.

Prompt 1:
Let's generate a three-unknown math word problem and its according system of equations. First, expand the two-unknown system of equations to three-unknown. Then, transfer the system of equations into according math word problem.
System of Equations:
0.01 * 35.0 * x + 0.01 * 15.0 * y = 0.01 * 27.0 * (200.0)
y = (200.0 - x)

Response 1:
Expanded System of Equations:
0.01 * 35.0 * x + 0.01 * 15.0 * y + 0.01 * 25.0 * z = 0.01 * 27.0 * 200.0
y + z = 200.0 - x
x - y = z - 50.0
Math Word Problem:
A company is investing in three different projects. The first project is expected to yield a 35% return, the second a 15% return, and the third a 25% return. The total investment is $200 and the total expected return is 27% of the total investment. If the company invests $50 more in the first project than in the third, and the total investment in the second and third projects equals the remaining amount after investing in the first project, how much is invested in each project?
Source Problem:
An alloy containing 15 % brass is to be combined with an alloy containing 35 % brass to form an alloy containing 27 % brass. How much of each alloy should be combined to make 200 pounds of the 27 % brass alloy?

Prompt 2:
Let's generate a three-unknown math word problem and its according system of equations. First, expand the two-unknown system of equations to three-unknown. Then, transfer the system of equations into according math word problem.
System of Equations:
6.0 * 0.01 * x + 9.0 * 0.01 * y = 684.0
x + y = 10000.0

Response 2:
Expanded System of Equations:
6.0 * 0.01 * x + 9.0 * 0.01 * y + 5.0 * 0.01 * z = 684.0
x + y + z = 10000.0
x - y = 2000.0
Math Word Problem:
A company is investing in three different projects. The first project is expected to yield a 6% return, the second a 9% return, and the third a 5% return. The total investment is $10,000 and the total expected return is $684. If the company invests $2000 more in the first project than in the second, how much is invested in each project?
Source Problem:
Suppose you invested 10,000 dollars, part at 6 % annual interest and the rest at 9 % annual interest. If you received 684 dollars in interest after one year, how much did you invest at 6 % annual interest and at 9 % annual interest?

Prompt 3:
Let's generate a three-unknown math word problem and its according system of equations. First, expand the two-unknown system of equations to three-unknown. Then, transfer the system of equations into according math word problem.
System of Equations:
7.0 * x + 5.0 * y = 146.0
x + y = 24.0

Response 3:
Expanded System of Equations:
7.0 * x + 5.0 * y + 3.0 * z = 146.0
x + y + z = 24.0
x - y = 2.0
Math Word Problem:
A store sells three types of fruits: apples, bananas, and cherries. Apples cost $7 per pound, bananas cost $5 per pound, and cherries cost $3 per pound. The store sold a total of 24 pounds of these fruits and made $146. If the store sold 2 more pounds of apples than bananas, how many pounds of each type of fruit were sold?
Source Problem:
Samuel has found that his new car gets 7 miles per gallon on the highway and 5 miles per gallon in the city. He recently drove 146 miles on 24 gallons of gasoline. How many miles did he drive on the highway? How many miles did he drive in the city?

Prompt 4:
Let's generate a three-unknown math word problem and its according system of equations. First, expand the two-unknown system of equations to three-unknown. Then, transfer the system of equations into according math word problem.
System of Equations:
5.0 + y = x
4.0 * x + 3.0 * y = 97.0

Response 4:
Expanded System of Equations:
5.0 + y + z = x
4.0 * x + 3.0 * y - z = 97.0
x - y = z + 2.0
Math Word Problem:
Three friends, Alice, Bob, and Charlie, are playing a game. The score of Alice is 5 more than Bob's score. When Alice's score is multiplied by 4 and added to 3 times Bob's score, it equals 97 minus Charlie's score. If Alice's score minus Bob's score equals Charlie's score plus 2, what are the individual scores of Alice, Bob, and Charlie?
Source Problem:
One number is 5 more than another number. Four times the larger number plus 3 times the smaller is 97. Find the numbers.

Table 19: Examples of multiple unknown algebra problem generation, corresponding system of equations and source problem using equation-aware prompt on GPT-4.

Case 1: How much 1 % boric acid solution, 5 % boric acid solution, and 2% boric acid solution are needed to make 40 mL of a solution that is 1.1% boric acid? The sum of the amounts of the 5% solution and the 2% solution equals the difference between 40 mL and the amount of the 1% solution. Also, the sum of the amounts of the 1% solution and the 5% solution equals twice the amount of the 2% solution plus 10 mL. **System of Equations:** (0.01 * 1.0) * x + (0.01 * 5.0) * y + (0.01 * 2.0) * z = 1.10 (40 * 0.0275) y + z = 40.0 - x x + y = 2.0 * z + 10.0 **Correct Question:** How much 1 % boric acid solution, 5 % boric acid solution, and 2% boric acid solution are needed to make 40 mL of a solution that is **2.75%** boric acid? The sum of the amounts of the 5% solution and the 2% solution equals the difference between 40 mL and the amount of the 1% solution. Also, the sum of the amounts of the 1% solution and the 5% solution equals twice the amount of the 2% solution plus 10 mL.
Case 2: If the price of copper is 65 cents per pound, the price of zinc is 30 cents per pound, and the price of nickel is 50 cents per pound, how many pounds of copper, zinc, and nickel should be mixed to make 80 pounds of an alloy selling for 45.625 cents per pound? The sum of the pounds of zinc and nickel equals 80 minus the pounds of copper. **System of Equations:** 65.0 * x + 30.0 * y + 50.0 * z = 3650.0 y + z = 80.0 - x x + y + z = 80.0 **Correct Question:** If the price of copper is 65 cents per pound, the price of zinc is 30 cents per pound, and the price of nickel is 50 cents per pound, how many pounds of copper, zinc, and nickel should be mixed to make 80 pounds of an alloy selling for 45.625 cents per pound? **The sum of the pounds of zinc and nickel equals 7 times the pounds of copper. (create new math relation)**
Case 3: Three cars are travelling in opposite directions. After two hours, they are 372 km apart. The first car is the slowest, the second car travels 6 kph faster than the first car, and the third car travels 4 kph slower than the first car. Find the average speed of each car. **System of Equations:** 2.0 * (x + y + z) = 372.0 y = x + 6.0 z = x - 81.0 **Correct Question: Cannot Expand**
Case 4: Sam, Masc, and Alex are friends. Masc is 7 years older than Sam and Alex is 5 years older than Masc. The sum of their ages is 49. What are their ages? **System of Equations:** y + x + z = 37.0 y = x + 7.0 z = y - 7.0 **Correct Question:** Sam, Masc, and Alex are friends. Masc is 7 years older than Sam and Alex is **7 years younger** than Masc. The sum of their ages is **37**. What are their ages?

Table 20: Error cases of multiple unknown algebra problem generation, corresponding system of equations, and correct question.

E1: Fewer or More Equation in the Set

Input:
{Instruction}
{Demos}
Question: The Rocket Coaster has expanded and now has 20 cars: some hold 4 people, some hold 6 people, and some hold 2 people. Altogether, there is room for 82 people. It is also known that the number of 4-passenger cars plus the number of 6-passenger cars equals three times the number of 2-passenger cars. How many of each type of car are there?

Output:
The system of equations for the given word problem is as follows:
1- The question is asking for the number of each type of car.
2- Relevant information:
  - Total number of cars: 20
  - Total capacity: 82 people
  - Relationship between the number of cars: 4-passenger + 6-passenger = 3 * 2-passenger
3- Assign symbols:
  Let a = number of 4-passenger cars
  Let b = number of 6-passenger cars
  Let c = number of 2-passenger cars
4- Mathematical relationships:
  - 4a + 6b + 2c = 82 (total capacity)
  - a + b = 3c (relationship between the number of cars)
5- System of equations:
  4a + 6b + 2c = 82
  a + b = 3c

E2: Wrong equation (meaning the coefficients or constant terms are wrong)

Input:
{Instruction}
{Demos}
Question: In a family, there are 3 cars. In a particular week, the sum of the average miles per gallon obtained by the three cars is 75. The first car has consumed 40 gallons, the second car has consumed 20 gallons, and the third car has consumed 10 gallons. This resulted in a total of 1700 miles driven by the three cars combined. Additionally, the sum of the average gas mileage of the first car and the third car is 15 more than twice the average gas mileage of the second car. What was the average gas mileage obtained by each of the 3 cars that week?

Output:
1- The question is asking for the average gas mileage obtained by each of the 3 cars in a particular week.
2- Relevant information:
  - Total sum of the average miles per gallon obtained by the three cars: 75
  - Gallons consumed by the first car: 40
  - Gallons consumed by the second car: 20
  - Gallons consumed by the third car: 10
  - Total miles driven by the three cars: 1700
  - Relationship between the average gas mileage of the cars
3- Assign symbols:
  Let a = average gas mileage of the first car
  Let b = average gas mileage of the second car
  Let c = average gas mileage of the third car
4- Mathematical relationships:
  - (40/a) + (20/b) + (10/c) = 75
  - 40a + 20b + 10c = 1700
  - (a + c) = 15 + 2b
5- System of equations:
  (40/a) + (20/b) + (10/c) = 75
  40a + 20b + 10c = 1700
  (a + c) = 15 + 2b

E3: Wrong Extraction Form of System of Equations

Input:
{Instruction}
{Demos}
Question: The Drama Club sold 1,700 tickets for the end-of-year-performance. Admission prices were 12 dollars for adults, 6 dollars for students, and 10 dollars for seniors. The total amount collected at the box office was 18,200 dollars. It is known that twice the number of student tickets plus the number of adult tickets equals three times the number of senior tickets plus 1200. How many students, adults, and seniors attended the play?

Output:
The system of equations for the given word problem is as follows:
1. Let a be the number of adult tickets sold, s be the number of student tickets sold, and r be the number of senior tickets sold.
2. Mathematical relationship:
  - a + s + r = 1700 (total tickets sold)
  - 12a + 6s + 10r = 18200 (total revenue collected)
  - 2s + a = 3r + 1200 (number of tickets relationship)
The system of equations is:
  a + s + r = 1700
  12a + 6s + 10r = 18200
  2s + a = 3r + 1200

Table 21: Different type of error example of Formulate-and-Solve on GPT-3.5.

Three-Unknown
Input: {Instruction} {Demos} Question: A chemist has three solutions: one that is 18 % alcohol, one that is 50 % alcohol, and one that is 10 % alcohol. He wants to make 100 liters of a solution that is 26 % alcohol. Also, the amount of the 18 % alcohol solution plus the amount of the 50 % alcohol solution he uses equals four times the amount of the 10 % alcohol solution. How many liters of each solution should he add?
Output: The system of equations for the given word problem is as follows: 1- The question is asking for the number of liters of each solution the chemist should add. 2- Relevant information: - Total liters of solution: 100 - Desired alcohol concentration: 26% - Relationship between the amount of solutions: amount of 18% solution + amount of 50% solution = 4 * amount of 10% solution 3- Assign symbols: Let a = liters of 18% alcohol solution Let b = liters of 50% alcohol solution Let c = liters of 10% alcohol solution 4- Mathematical relationships: - a + b + c = 100 (total liters of solution) - 0.18a + 0.50b + 0.10c = 0.26 * 100 (desired alcohol concentration) - a + b = 4c (relationship between the amount of solutions) 5- System of equations: a + b + c = 100 0.18a + 0.50b + 0.10c = 26 a + b = 4c
Four-Unknown
Input: {Instruction} {Demos} Question: In a laboratory, a chemist needs to prepare 300 milliliters of a 20 % HCl solution. She has available a 10 % HCl solution, a 30 % HCl solution, a 20 % HCl solution, and a 40 % HCl solution. She knows that the volume of the 30 % HCl solution plus twice the volume of the 10 % HCl solution equals five times the volume of the 20 % HCl solution plus 100 milliliters. The volume of the 30 % HCl solution plus the volume of the 10 % HCl solution plus the volume of the 20 % HCl solution equals four times the volume of the 40 % HCl solution plus 50 milliliters. How many milliliters of each solution should she use?
Output: The system of equations for the given word problem is as follows: 1- The question is asking for the volume of each solution to be used. 2- Relevant information: - Total volume needed: 300 milliliters - Mathematical relationships between the volumes of the solutions 3- Assign symbols: Let x be the volume of the 10% HCl solution. Let y be the volume of the 30% HCl solution. Let z be the volume of the 20% HCl solution. Let w be the volume of the 40% HCl solution. 4- Mathematical relationships: x + y + z + w = 300 (total volume needed) 0.1x + 0.3y + 0.2z + 0.4w = 0.2 * 300 (20% HCl solution) y + 2x = 5z + 100 (relationship 1) y + x + z = 4w + 50 (relationship 2) 5- System of equations: x + y + z + w = 300 0.1x + 0.3y + 0.2z + 0.4w = 60 y + 2x - 5z = 100 y + x + z - 4w = 50

Table 22: Full output example of Formulate-and-Solve on GPT-3.5 from three and four-unknown problem.

Five-Unknown

Input:

{Instruction}

{Demos}

Question: A chemist has five solutions: one that is 39 % alcohol, one that is 50 % alcohol, one that is 12 % alcohol, one that is 5 % alcohol, and one that is 3 % alcohol. He wants to make 140 liters of a solution that is 27.5 % alcohol. Also, the amount of the 39 % alcohol solution plus the amount of the 50 % alcohol solution he uses equals four times the amount of the 12 % alcohol solution. The amount of the 39 % alcohol solution plus the amount of the 50 % alcohol solution plus the amount of the 12 % alcohol solution equals five times the amount of the 5 % alcohol solution. The amount of the 39 % alcohol solution plus the amount of the 50 % alcohol solution plus the amount of the 12 % alcohol solution plus the amount of the 5 % alcohol solution equals six times the amount of the 3 % alcohol solution. How many liters of each solution should he add?

Output:

I can help you with that. Let's start by writing down the relevant information in simple statements:

1- The question is asking for the amount of each solution the chemist should add to make 140 liters of a solution that is 27.5% alcohol.

2- Relevant information:

- The chemist has five solutions: 39% alcohol, 50% alcohol, 12% alcohol, 5% alcohol, and 3% alcohol.
- He wants to make 140 liters of a solution that is 27.5% alcohol.
- The amount of the 39% alcohol solution plus the amount of the 50% alcohol solution he uses equals four times the amount of the 12% alcohol solution.
- The amount of the 39% alcohol solution plus the amount of the 50% alcohol solution plus the amount of the 12% alcohol solution equals five times the amount of the 5% alcohol solution.
- The amount of the 39% alcohol solution plus the amount of the 50% alcohol solution plus the amount of the 12% alcohol solution plus the amount of the 5% alcohol solution equals six times the amount of the 3% alcohol solution.

3- Assign symbols:

Let a = amount of 39% alcohol solution
Let b = amount of 50% alcohol solution
Let c = amount of 12% alcohol solution
Let d = amount of 5% alcohol solution
Let e = amount of 3% alcohol solution

4- Mathematical relationships:

a + b + c + d + e = 140
0.39a + 0.50b + 0.12c + 0.05d + 0.03e = 0.275 * 140
a + b = 4c
a + b + c = 5d
a + b + c + d = 6e

5- System of equations:

a + b + c + d + e = 140
0.39a + 0.50b + 0.12c + 0.05d + 0.03e = 38.5
a + b - 4c = 0
a + b + c - 5d = 0
a + b + c + d - 6e = 0

Table 23: Full output example of Formulate-and-Solve on GPT-3.5 from five-unknown problem.