

Reliable Reasoning Beyond Natural Language

Nasim Borazjanizadeh
University of California, Berkeley

Steven T. Piantadosi
Department of Psychology, UC Berkeley

Abstract

Despite their linguistic competence, Large Language models (LLMs) often exhibit limitations in their ability to reason reliably and flexibly. To address this, we propose a neurosymbolic approach that prompts LLMs to extract and encode all relevant information from a problem statement as logical code statements, and then use a logic programming language (Prolog) to conduct the iterative computations of explicit deductive reasoning. Our approach significantly enhances the performance of LLMs on the standard mathematical reasoning benchmark, GSM8k, and the Navigate dataset from the BIG-bench dataset. Additionally, we introduce a novel dataset, the Non-Linear Reasoning (NLR) dataset, consisting of 55 unique word problems that target the shortcomings of the next token prediction paradigm of LLMs and require complex non-linear reasoning but only basic arithmetic skills to solve. Our findings demonstrate that the integration of Prolog enables LLMs to achieve high performance on the NLR dataset, which even the most advanced language models (including GPT4) fail to solve using text only.

1 Introduction

The recent emergence of large language models (LLMs) [3, 25, 24, 8, 7, 29, 34, 35] has revolutionized the field of Natural Language Processing (NLP), with LLMs demonstrating human-level performance across various professional and academic benchmarks [26] and exhibiting an excellent understanding of linguistic rules and patterns [19].

However, despite their linguistic competence, LLMs often demonstrate significant limitations in their capacity to *reason* reliably and flexibly [19, 12, 39]. These limitations likely stem from the autoregressive architecture of transformers, which enforces the solution to the problems sequentially: the models’ reliance on a greedy process for predicting the next word constrains their backtracking and error recovery capability [12]. Models are expected to generate an answer in a single pass of their feedforward architecture, which cannot implement conditional loops [4]. Moreover, the statistical nature of LLMs’ training and representation means they often fail in generalizing appropriately to problems outside their training distribution, especially in settings requiring reasoning and discrete processes [40]. Furthermore, even the most advanced LLMs, including GPT4, have an incredibly short working memory [4], while reliable reasoning requires accurate and robust retrieval and integration of all relevant information.

Additionally, the linear and sequential nature of natural language contrasts with the complex and non-linear computations often involved in deductive reasoning. Even humans struggle with reasoning tasks when the brainstorming medium is confined to text. This is well illustrated by the history of logic. Aristotle’s writing on syllogistic reasoning, for example, lacked the tools of symbolic logic later developed for this kind of argumentation. The result is clunky and difficult to follow, even when correct:

If A has been proved to all or to some B, then B must belong to some A: and if
A has been proved to belong to no B, then B belongs to no A. This is a different

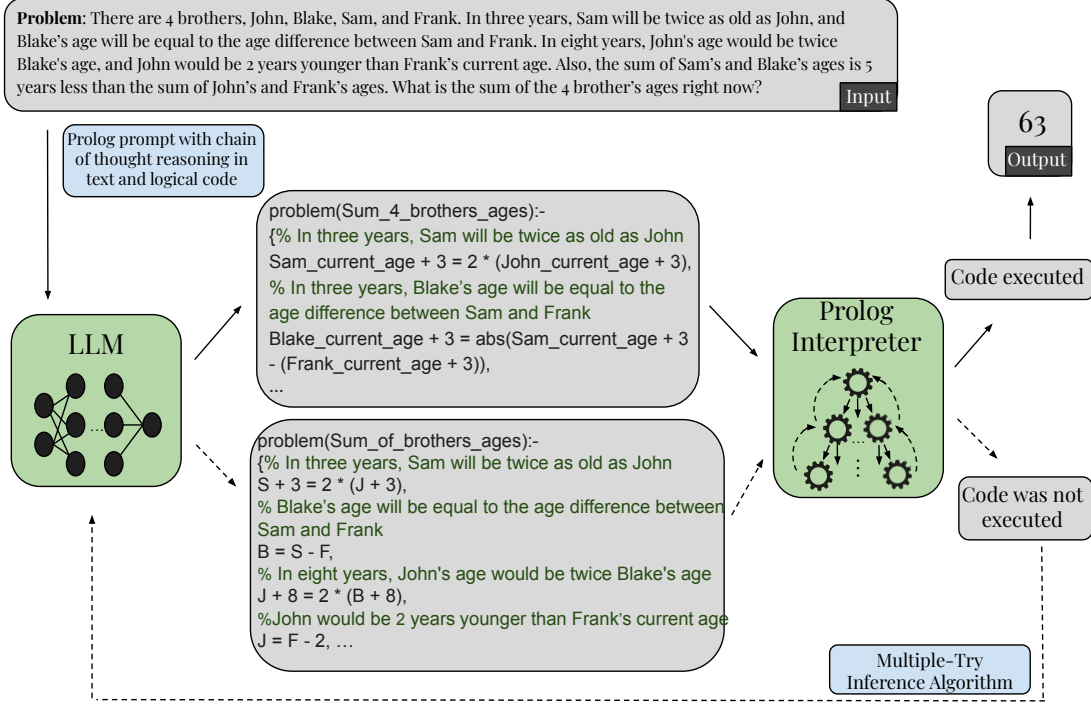


Figure 1: Our approach: A natural language problem (a math word problem from NLR dataset) is given to an LLM, which is prompted to perform CoT in text and logical code to encode the variable relationship as logical code statements. The Prolog interpreter executes the code. If the Prolog program fails, the LLM is re-prompted until valid code is generated or a limit of attempts is reached.

conclusion from the former. But if A does not belong to some B, it is not necessary that B should not belong to some A: for it may possibly belong to all A.

Others like Venn [36] and Boole [2] developed systems which allowed such reasoning to take place in a different medium—symbolic diagrams and algebraic equations respectively. These tools support reasoning about much richer types of logical relationships and deductive logic, than could be easily conveyed in natural language. Jevons [17] even developed a mechanical system for such logical reasoning, much in the spirit of Babbage’s work (see Gardner [14]). More generally, the principles and notation of mathematics allow us to concisely express concepts that would be incredibly difficult to express in natural language alone. Formalizing of reasoning in a system *other* than natural language has several descendants, from the General Problem Solver of Newell et al. [21], to logic programming languages like Prolog [11], and formal tools for robust verification like Lean [20]. Natural language is not enough for any of these domains.

2 Our Approach

To enable LLMs to perform deductive reasoning robustly, we propose integrating a reliable, deductive reasoning module into their inference pipeline. Specifically, in this study, we prompt the model to encode the constraints and relationships among variables, as described in the problem statement, as a set of Prolog code statements. The generated code is then evaluated by Prolog, which uses deductive approach, to derive a deterministic answer to the problem (Figure 1). This not only has the advantage of mirroring the likley human architecture of separate linguistic and reasoning systems [13, 19], but as we show, significantly improves the performance of LLMs in mathematical reasoning.

Indeed, this approach draws on the strengths of both symbolic and neural systems. Though systems like Prolog support reliable deduction, they have no mechanism to deal with the complexities and intricacies of natural language descriptions of problems. Moreover, they are unable to perform *implicit reasoning*, which involves extracting information that is not explicitly stated in the text but is

rather implied through common sense assumptions and context. However, Prolog and related systems excel at reasoning, with the ability to incorporate an arbitrary number of facts in their deductive processes, only generating valid conclusions given their assumptions. Prolog expresses knowledge as a set of relations, facts, and rules, and uses a reasoning engine to run queries over these relations, applying rules through resolution until a solution is found or all possibilities are exhausted. The ability to backtrack, conduct comprehensive searches, and accurately store and retrieve an arbitrary number of rules and relations are the capabilities that are difficult to implement using the feedforward architecture of LLMs, but essential for accurate deductive reasoning.

Moreover, in contrast to procedural or functional programming, declarative programming paradigm of Prolog focuses on defining what to execute and the program logic rather than specifying the detailed control flow. When LLMs are prompted to generate logical code to solve a problem, this declarative nature reduces the load on the LLM to define the variables or constraints encoded in the problem in the correct order or generate all intermediate steps of the computation correctly, allowing for a more direct mapping of the information encoded in natural language statements to logical code.

Two specific design choices help this approach work well. First, we prompt the LLM to perform **Chain of Thought (CoT) [38] reasoning in text and logical code**. This in-context learning method involves the integration of natural language comments that walk through the implicit reasoning steps required to arrive at the intermediate variables and code statements. While the code statements encode the explicit constraints and declarative arithmetic statements that the Prolog interpreter needs to compile. This technique allows the model to reason through the information implied by the context of problem statements and common sense but not explicitly stated (see e.g. Table 2). Second, we use the **Multiple Try** inference algorithm to obtain the models’ logical code generation for the problems. Using this inference method, if the Prolog code, generated by the LLM, fails to execute successfully¹, we rerun the model with a slightly increased temperature (with a preset maximum number of attempts) and return the numerical answer returned by the model’s first executable code generation (in contrast to, e.g., majority-vote schemes [37]). This approach helps to mitigate the brittleness of symbolic programming code.

We also introduce a novel dataset, the **Non-Linear reasoning dataset (NLR)** dataset², which is designed to evaluate the generalizability of LLMs’ mathematical reasoning capabilities. Motivated by the corruption of test and training sets for many mathematical tasks [26], and the simple and repetitive pattern of reasoning required to solve the problems of the current reasoning benchmarks [10, 31, 27, 9, 18], we present a new dataset that (i) is certainly outside of current models’ training sets, and (ii) each problem necessitates a unique and creative reasoning pattern to solve, while the mathematical skills needed are limited to basic arithmetic and algebra. This benchmark consists of unique constraint problems, math word problems, and problems that require following algorithmic instructions for updating a game model (see e.g. Table 1). We demonstrate that the most advanced LLMs, including GPT4, struggle to solve these problems when prompted to solve them step by step, utilizing a chain of thought text prompt, despite their success on other mathematical tasks [26, 4].

3 Comparison to Other Approaches

Several studies have explored the integration of LLMs with external tools and symbolic reasoning modules [23, 22, 10, 30]. For instance, training LLMs to make API calls to tools like calculators, interpreters, or external datasets has been shown to improve their performance across a variety of reasoning tasks [10, 30, 28]. While these methods have successfully reduce the arithmetic errors of LLMs, they do not sufficiently address the reasoning limitations inherent to the next-token prediction paradigm of LLMs and the linear nature of text, which can restrict the ability to perform comprehensive searches over the space of possibilities, explore multiple pathways to a solution, or backtrack.

Our approach builds upon and extends the work of LINC [23] and Nye et al. [22]. LINC uses a neurosymbolic process to convert natural language into first-order logic expressions with LLMs to determine the truth value of conclusions via a symbolic theorem prover. This method has shown significant performance gains on the FOLIO [15] and ProofWriter [33] datasets compared to CoT

¹This primarily occurs due to variable name assignment errors, as data flows are described without mutability in Prolog’s declarative syntax, unlike procedural programming.

²Link to NLR dataset

NLR Problem Statement	Characteristics
Math Word Problem: When I was half my current age, my father was 30. When I was $\frac{1}{3}$ my current age, my mother was 25. And when I was $\frac{1}{6}$ of my current age, my sister was 7. If the sum of my age, my sister’s age, my father’s age, and my mother’s age is 116, then how old am I now?	4 entangled variables in the problem
Constraint Satisfaction: In a line to enter a cinema, 4 people are standing between Bob and Alex. Chad’s index in the line is 1 after Bob’s, he’s standing right behind Bob considering the order of people left to right. Frank is right behind Alex. Sam is right in front of Bob. There are 2 people between Sam and Frank. If Bob is in the 7th person in the line, counting left to right, what is the number of Alex?	2 constraints encoding multiple possibilities
Algorithmic Instructions: There’s a cinema with 12 seats organized in 3 rows and 4 columns. Due to covid there’s a policy that a seat can be filled only if none of the seats right next to it in the same column or the same row are not filled. If we place a person in the seat in the second column of the first row and then start to fill the seats left to right, row by row, starting row with 1, how many people can be seated in the cinema in total?	5 entangled variables in each state

Table 1: Examples of each problem category in the NLR dataset

prompting. However, it has a limitation in capturing implicit information not explicitly stated in the premises, as it primarily uses LLMs as a semantic parser, translating each natural language premise directly into a logical statement [23]. Similarly, Nye et al. [22] improves the performance of LLMs in story generation and instruction-following tasks by using a symbolic reasoning module to check the logical consistency of generated text against a minimal world model. This method increases accuracy and robustness of neural generation but is limited by the need for hand-crafting the world model and defining specific constraints.

In our approach, the world model is constructed by the LLM itself, with no limitations on the number of constraints that can be encoded in the problem. Moreover, rather than using LLMs as semantic parsers or text-to-logical code translators, we prompt the LLM to perform chain of thought (CoT) reasoning in both text and logical code, prompting the LLM to conduct implicit reasoning and use additional tokens as working memory to derive intermediate variables. This enables a more flexible and generalizable reasoning process, making our neurosymbolic approach applicable to a wider variety of problems.

Our approach is also similar to the ‘Program of Thought’ (PoT) method [5], which separates computation from reasoning and language understanding. In PoT, LLMs generate text (as comments) and programming language statements to solve problems, delegating computation to a program interpreter. However, PoT’s code statements often directly translate the comments. In contrast, our in-context prompts use comments to walk through implicit reasoning steps, making them an integral part of the CoT. This allows the comments to encode different reasoning from the code statements, extracting necessary but unstated information to generate the correct logical code. (see Table 2 for examples of implicit reasoning performed by GPT4 in the comments).

4 Experiments

We first present results of our approach on two existing datasets: the standard mathematical reasoning dataset, GSM8k, and the Navigate task, extracted from the BIG-bench benchmark. In our experiments, we compare our approach against the standard prompting method used for reasoning tasks: examples solved using Chain of Thought (CoT) reasoning in text.

GSM8k [10] is a widely used benchmark for mathematical reasoning tasks, comprising of elementary school math problems. We tested the performance of our approach using GPT4 and GPT3.5 Turbo (hereafter GPT3.5), and text-davinci-003 models on the GSM8k dataset. To construct the Prolog prompt, we selected eight problems from the first 25 problems in the shuffled test split of the dataset. This selection was made to ensure that the prompt examples covered a variety of difficulty levels (The GSM8k dataset does not provide a difficulty score for the problems). The variable names in the Prolog prompts include a brief description and the unit of the variable in order to provide additional

<p>Problem: I am a 4 digit number. My rightmost digit is not divisible by 2. The sum of my digits is 20, and all my digits are in strictly decreasing order from left to right. One of my digits is 4 times one of my other digits, and the difference between my 2 middle digits is more than 3. What number am I? Answer: 9821</p>	
<pre> problem(Number):- % The number is a 4 digit number. We can % write it in terms of its digits: Number #= 1000 * Digit4 + 100 * Digit3 + 10 * Digit2 + Digit1, % Each of the digits of the number are % single digit numbers in range [0, 9] Digit1 #>= 0, Digit1 #< 10, Digit2 #>= 0, Digit2 #< 10, Digit3 #>= 0, Digit3 #< 10, % ** Besides the leftmost digit which can % not be 0 because otherwise the number % would be a 3 digit number. ** The % leftmost number must be in range [1, 9] Digit4 #> 0, Digit4 #< 10, % The rightmost digit is not divisible by 2 Digit1 mod 2 #\= 0, % The sum of the digits is 20 Digit1 + Digit2 + Digit3 + Digit4 #= 20, % All the digits are in strictly decreasing % order from left to right Digit4 #> Digit3, Digit3 #> Digit2, Digit2 #> Digit1, % One of the digits is 4 times one of the % other digits. ** There are 12 % possibilities for this condition, as % there are 12 unique digit pairs in this % number **: (4 * Digit1 #= Digit2; 4 * Digit1 #= Digit3; 4 * Digit1 #= Digit4; 4 * Digit2 #= Digit1; 4 * Digit2 #= Digit3; 4 * Digit2 #= Digit4; 4 * Digit3 #= Digit1; 4 * Digit3 #= Digit2; 4 * Digit3 #= Digit4; 4 * Digit4 #= Digit1; 4 * Digit4 #= Digit2; 4 * Digit4 #= Digit3), % The difference between the 2 middle % digits is more than 3 abs(Digit3 - Digit2) #> 3. </pre>	<p>Let's denote the four digits of the number as a, b, c, and d (from left to right).</p> <p>From the problem, we know that:</p> <ol style="list-style-type: none"> 1. d is not divisible by 2, which means d is an odd number. The possible values for d are 1, 3, 5, 7, 9. 2. $a + b + c + d = 20$, which means the sum of the four digits is 20. 3. The digits are in strictly decreasing order from left to right, which means $a > b > c > d$. 4. One of the digits is 4 times one of the other digits. 5. The difference between the two middle digits (b and c) is more than 3, which means $b - c > 3$. <p>Let's start solving the problem:</p> <p>Since the digits are in strictly decreasing order and the sum of the digits is 20, a must be at least 9. If a is 9, then $b + c + d = 11$. Since $b > c > d$ and d is odd, the maximum value for d is 5 (because if d is 7 or 9, then $b + c$ would be less than 4, which is not possible because b and c must be different and greater than d).</p> <p>So, if d is 5, then $b + c = 6$. The only possible values for b and c are 4 and 2, but the difference between b and c is not more than 3. So, d cannot be 5.</p> <p>If d is 3, then $b + c = 8$. The possible values for b and c are 5 and 3, but the difference between b and c is not more than 3. So, d cannot be 3.</p> <p>If d is 1, then $b + c = 10$. The possible values for b and c are 7 and 3, and the difference between b and c is more than 3. Also, one of the digits is 4 times one of the other digits (3 is 4 times 1). So, the number is 9731.</p>

Table 2: Comparison of GPT4's CoT reasoning in text vs in text and logical code for a constraint satisfaction problem in the NLR dataset. Red text indicates incorrect reasoning steps. **, added manually, encloses implicit reasoning steps in the comments. In the above example compiling the Prolog code generated by the model results in the correct answer.

information about feasibility of an operation between two variables. We used the CLP(R) library in Prolog [1] to write the Prolog code to enable declarative arithmetic and solving systems of linear equations.

Figure 2.a demonstrates that integrating Prolog with text-davinci-003 and GPT3.5 models significantly improved their performance on GSM8k, highlighting the benefit of incorporating a reliable reasoning module in the inference pipeline of LLMs. The declarative nature of Prolog facilitates outlining the program logic, thus reducing the load on the LLM by eliminating the need to specify the control flow, order of the operations, or accurately generate all intermediate computational steps. The performance of GPT4 also showed considerable improvement with the integration of Prolog. It's important to note

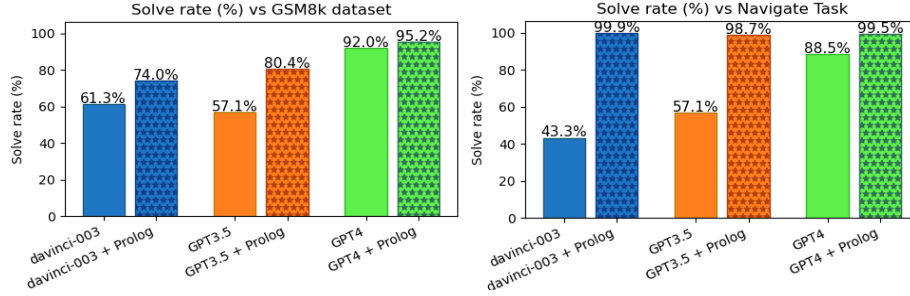


Figure 2: Comparing single model accuracy on GSM8k and Navigate benchmarks using text-only CoT versus our neurosymbolic approach (GPT3.5 + Prolog, GPT4 + Prolog), using CoT in text and logical code and the multiple-try inference algorithm. Few-shot CoT in text baselines on GSM8k references are Bubeck et al. [4] and OpenAI [26]

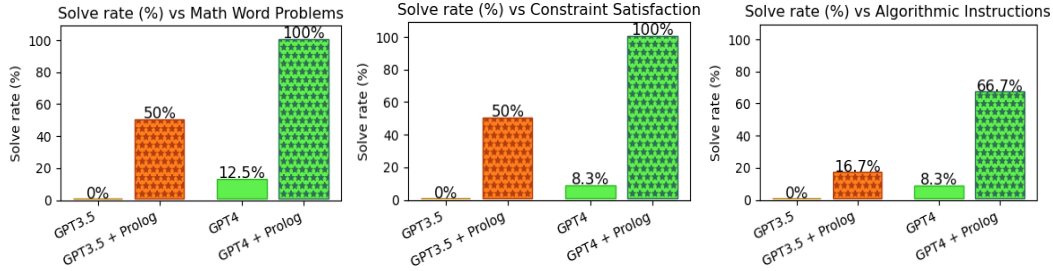


Figure 3: Comparing single model accuracy of LLMs (GPT3.5, GPT4) on the NLR dataset when prompted with text-only CoT versus our neurosymbolic approach (GPT3.5 + Prolog, GPT4 + Prolog), using CoT in text and logical code and the multiple-try inference algorithm.

that GPT4 is trained on the GSM8k training dataset during its pre-training phase [26], leading to its higher performance when prompted with CoT in text.

The **Navigate** task is a component of the BIG-bench benchmark, a collection of tasks designed to evaluate the language understanding and generation capabilities of LLMs [32]. This task specifically assesses the LLMs’ ability to solve a simple spatial reasoning task, involving tracking an agent’s location based on instructions detailing number of steps and the direction. The task requires iteratively updating a world model where each state has a few variables, the x and y coordinates and the directions the agent is facing.³ Navigate, like many other benchmark datasets such as RuleTaker [9], is systematically generated. The problem statements are constructed by sampling a combination of instructions, among a pool of nine instructions, with a random number of steps added to each instruction. Originally, the Navigate task requires determining whether the agent returns to its starting location, inherently introducing a 50% chance of correctness due to the binary nature of the answer. To alleviate this, we revised the task to ask the final distance of agent from the start. To construct the Prolog prompt, we used two examples of each of the two types of problems in this dataset. Since there is no reported text-based baseline for the GPT models for the Navigate task, we prompted the models with CoT in text to establish the baseline performance for this task. For comparability, we constructed a four-shot CoT in text prompt with the same four problems that were used to build the Prolog prompt.

Figure 2.b demonstrates a performance exceeding 98% for all three models integrated with Prolog on the Navigate task, which is a significant improvement compared to the performance of models prompted with CoT in text. This data suggests that the integration of a symbolic reasoning module can help prevent arithmetic errors in LLMs and assist with the models’ limited working memory when tracking and updating variables of the world model states. Notably, the davinci-text-003 and GPT3.5 models, which have a more restricted working memory relative to GPT4, showed the most significant improvements.

³We selected Navigate for our experiments among other BIG-bench tasks due to the fact that PaLM 540B and PaLM 62B’s performance on the task were significantly below the best human performance score [6].

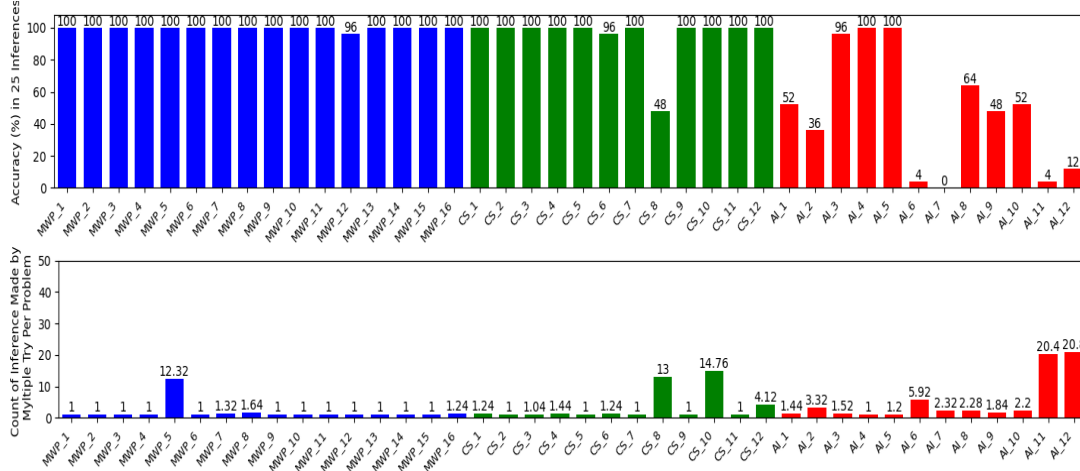


Figure 4: Evaluating the robustness and performance variability of our best model, GPT4 + Prolog, by running it 25 times on each NLR problem and recording the accuracy and average number of attempts it took to generate a valid code using the multiple-try inference algorithm. Math Word Problems are abbreviated as 'MWP', Constraint Satisfaction as 'CS', and Algorithmic Instruction as 'AI'.

5 NLR Dataset

We next introduce a new dataset, the **Non-Linear Reasoning dataset** (NLR), a collection of 55 problems hand-designed to evaluate the generality of LLMs' mathematical reasoning capabilities to out-of-distribution problems requiring complex, non-linear reasoning. The NLR dataset contains three categories of problems: (i) Math word problems (21 problems), (ii) Constraint satisfactions problems (17 problems), and (iii) Algorithmic instructions for updating a game model (17 problems).

Unlike other reasoning datasets that are syntactically generated, where the problem statements lack diversity in the wording, and premises and conclusions are expressed in short, direct statements [15, 9], our problems necessitate richer natural language understanding and both implicit and explicit deductive reasoning for resolution (see e.g. Table 2). The problems are also designed to disrupt the statistical co-occurrence patterns of words, by drawing inspiration from popular logic problems, but uniquely modifying the rules/constraints of the problems.

In contrast to the MATH dataset [16], which requires advanced mathematical skills such as calculating integrals and eigenvalues, the NLR problems only require basic arithmetic. However, despite the simplicity of the required math operations, LLMs struggle to solve NLR problems end to end (as shown in Figure 3). This is likely due to the high interrelationship between variables of the problem and the requirement to store and backtrack to intermediate states and trace multiple possible paths to the solution, which is challenging to perform using LLM's next word prediction paradigm.

To better understand the performance of the models on the NLR dataset, we introduce a notion of "*entanglement*" between variables in a problem. Variables are entangled when the value of a variable must be inferred through its relationship with other variables and modifying the value of the variable affects the value of the variables entangled with it. For instance, in the math word problem presented in Table 1, each person's age is defined in terms of other people's age, rather than by a specific numerical value. Consequently, the system of equations encoding the relationships of the variables require a nonlinear computational process for resolution, which involves an iterative process of simplifying and substituting more complex linear equations. The algorithmic instruction problem in Table 1 also exhibit variable entanglement. Seating a person in the cinema affects the availability of the 4 adjacent seats. Hence, the seat value is entangled with the values of the 4 neighboring seats. Consequently, all variables in the game model's current state must be updated based on the problem's conditionals and update rules after each action. This significantly challenges the model's working memory.

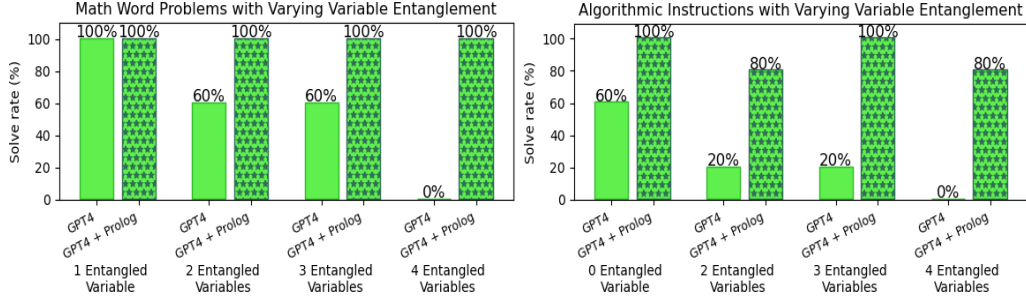


Figure 5: Comparing single model accuracy of GPT4 using a text-only CoT prompt versus our neurosymbolic approach on variations of a subset of NLR problems with 0-4 entangled variables.

To illustrate the significant impact of variable entanglement on the model’s performance, we designed instances with similar structure and reasoning pattern but varying numbers of entangled variables for five math word problems and five algorithmic instruction problems in the NLR dataset⁴⁵. Figure 5 shows that the capacity of GPT4 to solve the problems end-to-end drastically declines as the number of entangled variables in the problem increases, with the model failing to solve any of the problems that have four entangled variables when prompted with the standard CoT in text. In contrast, using our neurosymbolic approach, the model maintains a consistently strong performance across all of the problems.

We conducted two more series of experiments on the NLR dataset, using GPT3.5 Turbo and GPT4⁶. The first compares the mean performance of our model against the text-only CoT prompting baseline across all problems (Figure 3). The second assesses our model’s robustness and performance variability by running it 25 times on each NLR problem, recording the accuracy and the average number of attempts needed to produce valid code for each problem using the multiple-try inference algorithm (Figure 4). We used a five-shot prompt of CoT in text and logical code, and a five-shot text-only prompt for each problem category. The text-only prompts were generated by initially instructing GPT-4 to solve the problems step-by-step, 0-shot, and then debugging the solutions. We conducted inference on a total of 40 problems (16 math word problems, 12 constraint satisfaction problems, and 12 algorithmic instructions problems), using the multiple-try algorithm for the Prolog augmented inference pipeline.

(i) - Math Word Problems With Variable Entanglement

As shown in Figure 3.a, the simple modification of defining variables in relation to other variables significantly impacts the end-to-end performance of LLMs in solving math word problems. While GPT4 is generally successful in extracting the equations representing the relationship between the entangled variables from the textual information, it struggles to solve the resulting system of linear equations when prompted with CoT in text (contrasting strongly with its almost perfect performance on the GSM8k dataset). This suggests that while LLMs are adept at understanding the semantic meaning of problem statements, they struggle with the non-linear computations of solving a system of linear equations, even though the mathematical scope of operations is limited to simple algebra.

As evidenced by the 100% accuracy we obtained by integrating GPT4 with Prolog, prompting the model to define the variable relationships as equalities in a Prolog predicate and utilizing Prolog’s declarative arithmetic to solve the resulting system of equations, eliminates the limitation on the number of entangled variables (Figure 5) and allows the models to solve these problems robustly. This is evidenced by our second experiment, (Figure 4), where our model successfully solved nearly all math word problems in all 25 attempts, with very few inference attempts.

⁴Link to the NLR dataset variation with varying number of entangled variables

⁵Algorithmic instructions and math word problems in the original NLR dataset have 3-5 entangled variables.

⁶The API to the text-davinci-003 model was disabled at the time we ran the experiments on the NLR dataset

(ii) - Constraint Satisfaction Problems

Figure 3.b demonstrates that the LLMs mostly fail to solve the NLR constraint satisfaction problems when prompted with CoT in text. The models often overlook some possibilities, hallucinate about whether a possibility satisfies all constraints, or make illogical leaps in reasoning which in turn results in a 8% success rate in solving these problems. For instance, consider the constraint satisfaction problem presented in Table 1, which states that there are four people between Bob and Alex in a line. The LLMs only consider the possibility where Bob is standing in the i th index and Alex in $(i + 5)$ th index of the line, which is one of the two possible orders of Bob and Alex. The existence of constraints that encode multiple possibilities makes these problems particularly difficult for LLMs due to the extensive non-linear reasoning required to trace and revisit all potential solutions.

However, as demonstrated by Figure 3, prompting GPT4 to formulate the information encoded in the problems as Prolog predicates, where the model’s task is to encode the constraints as logical code statements rather than attempting to iteratively check the possible states against the constraints, effectively addresses these issues. This approach resulted in a 100% success rate in solving the problems. GPT3.5, on the other hand, succeeds in correctly encoding the constraints as logical code in Prolog half of the time, which demonstrates that GPT3.5 is less proficient than GPT4 in inferring the information implied by the natural language statements of the problems.

(iii) - Algorithmic Instructions for Updating a Game Model

The algorithmic instructions problems in the NLR dataset are designed to evaluate the ability of LLMs to implement the algorithm described in the problem statement to track and update the states of a world model. Similar to the Navigate task, these problems provide a deterministic set of instructions for updating the given initial state of the world, leading to a guaranteed feasible end state. Thus, in solving these problems, the LLMs are not required to make decisions about the choice or order of actions. However, in contrast to the Navigate task, where the problems are systematically generated instances of the same task, each problem in the NLR dataset defines a new task with unique rules and reasoning.

As shown in Figure 3.3, GPT4 solved 66.7% of the problems with the integration of Prolog, a significant improvement compared to the 8% success rate achieved by the model when solving these problems using text only. When tasked to solve the problems end-to-end, the models struggle to accurately store and retrieve the intermediate states, which are typically lists of length 10, often omit numbers when rewriting the updated list, fail to consider all of the conditionals in the given algorithm when updating a state, or fail to apply the algorithm for the correct number of steps.

Figure 4 illustrates the correlation between the model’s robustness, accuracy, and the reasoning required to map the information encoded by the problem statements into Prolog code. Despite notable improvement over the text-only baseline in solving algorithmic instructions problems, our model is least stable in solving this category, both in terms of accuracy and average number of inference attempts made before the model generated a valid code. This is anticipated as these tasks mainly involve modifying a game state, which does not necessitate deductive reasoning, rendering Prolog’s reasoning engine less beneficial, and placing more load on the LLM to write the logical code that initializes and updates a data structure that represents the world model.

6 Limitations and Broader Impact

Our work, which is on the path to creating models that can reason generally, reliably, and correctly like humans, can lead to several positive societal impacts. Such models can significantly increase the reliability and usefulness of current AI. In turn, improving the reasoning capabilities of language models could lead to job displacement as they can more reliably automate complex tasks traditionally performed by humans.

The main limitation of the NLR dataset is scalability. Designing tasks that require unique reasoning patterns for resolution, math word problems with high variable entanglement, constraint satisfaction problems with constraints encoding multiple possibilities, and game algorithms with new rules is complex and time consuming. Moreover, LLMs may produce coherent but incorrect logical code solutions, making error detection challenging. Additionally, Prolog’s limited infrastructure support

for complex data structures, compared to languages like Python, may restrict its applicability to problems involving higher-dimensional data.

7 Conclusion

This work highlights inherent limitations of LLMs in performing reliable and generalizable reasoning, and suggests these problems can be mitigated by integrating a symbolic reasoning system into their inference pipeline. Our neurosymbolic approach prompts the LLM to map the information encoded by problem statements to logical code statements, thereby delegating the iterative computations of explicit deductive reasoning to a reliable reasoning engine. This division of labor significantly enhances the performance of LLMs on mathematical reasoning tasks. Additionally, our novel NLR dataset provides a robust benchmark for evaluating the generality of LLMs’ mathematical reasoning capabilities to problems that require unique nonlinear reasoning and challenge the limitations of the linear next word prediction paradigm of LLMs.

References

- [1] URL <https://www.swi-prolog.org/man/clpqr.html>.
- [2] George Boole. *The laws of thought*, volume 2. Open court publishing Company, 1854.
- [3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [4] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.
- [5] Wenhui Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.
- [6] 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 Sankaranarayanan 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. Palm: Scaling language modeling with pathways, 2022.
- [7] 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 M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. 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 García, 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 Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm:

- Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113, 2022. URL <https://api.semanticscholar.org/CorpusID:247951931>.
- [8] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.
 - [9] Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language, 2020.
 - [10] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
 - [11] Alain Colmerauer and Philippe Roussel. The birth of prolog. In *History of programming languages—II*, pages 331–367. 1996.
 - [12] Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36, 2024.
 - [13] Evelina Fedorenko and Rosemary Varley. Language and thought are not the same thing: evidence from neuroimaging and neurological patients. *Annals of the New York Academy of Sciences*, 1369(1):132–153, 2016.
 - [14] Martin Gardner. *Logic Machines and Diagrams*. University of Chicago Press, Chicago, 1958.
 - [15] Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, et al. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*, 2022.
 - [16] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.
 - [17] William Stanley Jevons. I. on the mechanical performance of logical inference. *Proceedings of the Royal Society of London*, 18(114-122):166–169, 1870.
 - [18] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*, 2017.
 - [19] Kyle Mahowald, Anna A Ivanova, Idan A Blank, Nancy Kanwisher, Joshua B Tenenbaum, and Evelina Fedorenko. Dissociating language and thought in large language models: a cognitive perspective. *arXiv preprint arXiv:2301.06627*, 2023.
 - [20] The mathlib Community. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, POPL ’20*. ACM, January 2020. doi: 10.1145/3372885.3373824. URL <http://dx.doi.org/10.1145/3372885.3373824>.
 - [21] Allen Newell, John C Shaw, and Herbert A Simon. Report on a general problem solving program. In *IFIP congress*, volume 256, page 64. Pittsburgh, PA, 1959.
 - [22] Maxwell Nye, Michael Tessler, Josh Tenenbaum, and Brenden M Lake. Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning. *Advances in Neural Information Processing Systems*, 34:25192–25204, 2021.
 - [23] Theo X Olausson, Alex Gu, Benjamin Lipkin, Cedegao E Zhang, Armando Solar-Lezama, Joshua B Tenenbaum, and Roger Levy. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. *arXiv preprint arXiv:2310.15164*, 2023.

- [24] OpenAI. Chatgpt: Optimizing language models for dialogue, 2022. URL <https://openai.com/index/chatgpt/>.
- [25] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023. URL <https://api.semanticscholar.org/CorpusID:257532815>.
- [26] R OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023.
- [27] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.
- [28] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*, 2023.
- [29] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John F. J. Mellor, Irina Higgins, Antonia Creswell, Nathan McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, L. Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, N. K. Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Tobias Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d’Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew G. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William S. Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem W. Ayoub, Jeff Stanway, L. L. Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher. *ArXiv*, abs/2112.11446, 2021. URL <https://api.semanticscholar.org/CorpusID:245353475>.
- [30] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.
- [31] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.
- [32] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubakaran, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, César Ferri Ramírez, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta,

Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodola, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Wang, Gonzalo Jaimovitch-López, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Shevlin, Hinrich Schütze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Kocoń, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh D. Dhole, Kevin Gimpel, Kevin Omond, Kory Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros Colón, Luke Metz, Lütfti Kerem Şenel, Maarten Bosma, Maarten Sap, Maartje ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramírez Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael A. Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michał Śwędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimeo Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T, Nanyun Peng, Nathan A. Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan LeBras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Ruslan Salakhutdinov, Ryan Chi, Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel S. Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima, Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven T. Piantadosi, Stuart M. Shieber, Summer Mishnerghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsu Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai,

- Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2023.
- [33] Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *arXiv preprint arXiv:2012.13048*, 2020.
 - [34] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.
 - [35] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.
 - [36] John Venn. *Symbolic Logic*. 1881.
 - [37] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
 - [38] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.
 - [39] Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. *arXiv preprint arXiv:2307.02477*, 2023.
 - [40] Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang, and Guy Van den Broeck. On the paradox of learning to reason from data. *arXiv preprint arXiv:2205.11502*, 2022.