MATHGAP: OUT-OF-DISTRIBUTION EVALUATION ON PROBLEMS WITH ARBITRARILY COMPLEX PROOFS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) can solve arithmetic word problems with high accuracy, but little is known about how well they generalize to problems that are more complex than the ones on which they have been trained. Empirical investigations of such questions are impeded by two major flaws of current evaluations: (i) much of the evaluation data is contaminated, in the sense that it has already been seen during training, and (ii) benchmark datasets do not capture how problem proofs may be arbitrarily complex in various ways. As a step towards addressing these issues we present a framework for evaluating LLMs on problems with arbitrarily complex arithmetic proofs, called MathGAP. MathGAP generates problems that follow fixed proof specifications—along with chain-of-thought reasoning annotations—enabling systematic studies on generalization with respect to arithmetic proof complexity. We apply MathGAP to analyze how in-context learning interacts with generalization to problems that have more complex proofs. We find that among the models tested, most show a significant decrease in performance as proofs get deeper and wider. This effect is more pronounced in complex, nonlinear proof structures, which are challenging even for GPT-4o. Surprisingly, providing in-context examples from the same distribution as the test set is not always beneficial for performance. In particular, zero-shot prompting as well as demonstrating a diverse range of examples that are less complex than the test data sometimes yield similar or higher accuracies.

1 INTRODUCTION

High performance on reasoning benchmarks is often taken as evidence that transformer-based large language models (LLMs) can "reason". However, many current evaluations are unreliable since it is likely that the problems they contain are present in the model's training data (Sainz et al., 2023; Deng et al., 2024; Zhang et al., 2024). Moreover, most evaluations fail to capture that reasoning problems can be arbitrarily complex, through composition of subproofs and use of multiple rules of inference. High accuracy on a specific set of problems in a benchmark dataset is therefore not sufficient to conclude that LLMs can generalize to more complex, unseen problems. To obtain a more appropriate empirical measure of reasoning ability, one must evaluate on data that is *not* present in any benchmark dataset, containing proofs that are more complex than those used in training and context window alike.

Math word problems (MWPs) are one of the most frequently used testbeds for LLM reasoning. Yet there exist few, if any, systematic analyses on out-of-distribution (OOD) generalization in regards to proof complexity that avoid contamination in this domain. We suspect that this is in part due to a lack of formalism for characterizing proof complexity, which we argue is necessary for making precise claims about generalization behavior for reasoning problems.

In this paper, we present a framework for evaluating Mathematical Generalization on Arithmetic Proofs—**MathGAP**. First, we discuss how to characterize solutions to MWPs in terms of proof trees, in which the nodes are labeled with logical statements about the problem under a world-model framework (Opedal et al., 2023), and the edges are induced by proof steps on such logical statements. Doing so reveals several ways of characterizing a problem's complexity based on its proof tree, e.g., (non)linearity, depth, width, and the ordering of nodes (all defined in §3.2). With this formalism, we develop a method to generate problems with specified proof tree characteristics. Because this method is based on proof trees, it also enables rich annotations in the form of ground-truth reasoning traces. More specifically, traversing the proof trees in post order, we obtain chain-of-thought (CoT; Wei et al., 2022) reasoning traces in an automated manner. Fig. 1 illustrates the generation method, showing

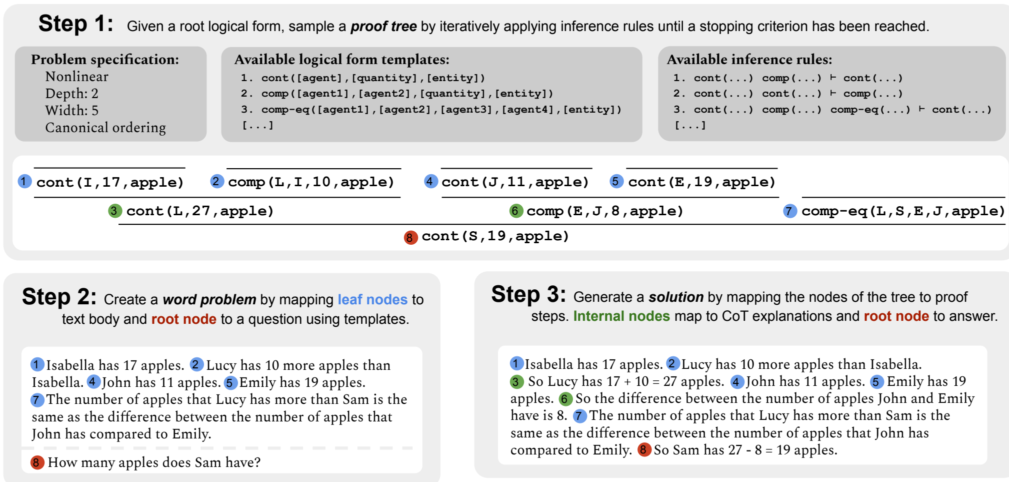

Figure 1: We propose MathGAP, an evaluation framework for arithmetic reasoning in which LLMs are tested on problems with proofs of arbitrary complexity. This diagram shows how problems and CoT solution annotations are generated under our formalism. The complete list of logical forms and inference rules that we consider in our experiments are given in Tables 1 and 2, respectively.

a nonlinear proof tree with depth 2 and width 5, in which the ordering of the sentences is given by traversing the leaf nodes in left-to-right order. This generation method forms the backbone of MathGAP, our framework for studying OOD generalization gaps on arithmetic proofs. MathGAP avoids data contamination since the distribution of generated test problems can always be different from those in training or in context—by design. When the performance on problems at one level of complexity hits saturation, we can flexibly generate a new set of problems that are even more complex. This is in spirit similar to Dynabench (Kiela et al., 2021) but, importantly, does not require human annotators.

MathGAP opens up the possibility for several new empirical investigations.[1] In the present study, we perform a systematic analysis on whether LLMs can use simple examples in context to solve more complex ones at inference. To this end, we consider two different distributions of in-context CoT demonstrations: examples with only the simplest possible proof trees, and a range of examples with varying complexity, akin to curriculum learning (Bengio et al., 2009). We then evaluate on test sets containing problems that are increasingly more complex than the in-context examples. The performance is compared to two baselines of in-context demonstrations, a prompt that has in-distribution examples with respect to the test set, and a zero-shot prompt. Our study includes Mixtral-8x7B (Jiang et al., 2024a), Llama3-8B, Llama3-70B (Llama Team, 2024), GPT-3.5-Turbo, GPT-4o (OpenAI, 2024), and a few additional evaluations on o1-preview. We present several findings:

- As expected, there is a consistent decrease in performance as proof depth and width increase. The decrease is particularly notable for nonlinear problems, for which we observe a general inability of the evaluated LLMs to solve complex problems, even for GPT-4o. Moreover, nonlinear problems are overall more difficult than linear problems when controlling for length.
- Generalization across orderings appears to be different, however. We conduct experiments where we move sentences to the front of the problem and find that the problem is easiest for LLMs if the sentence is moved from near the beginning or end, rather than from the middle of the problem. This is surprising since one would expect a monotonically decreasing relationship between accuracy and the distance of the movement.
- We see no clear relationship between in-context distribution and performance. Contrary to our expectations, in-distribution examples are not always beneficial for performance, as compared to zero-shot. This may be indicative of the training procedure, or alternatively, suggest that it is problem complexity rather than lack of OOD generalization that explains the decline in performance. Demonstrating a range of examples with varying complexity is usually preferable to demonstrating simple examples, suggesting that LLMs benefit from a diverse prompt.
- The o1-preview model outperforms the other models on nonlinear problems. Yet, we are able to create a test set of which it solves only 5%—demonstrating that MathGAP can be future-proof.

[1] Code to generate problems with MathGAP will be publicly released with the final version.

2 Related Work

Data contamination. Machine learning, broadly, is the study of algorithms that can learn from data, and generalize to *unseen* data. Machine learning models must therefore be evaluated on unseen test sets. As it turns out, modern-day LLMs are not being properly evaluated in this regard: Much of the data they are evaluated on has already been used during model training, both labeled (Dodge et al., 2021; Deng et al., 2024) and unlabeled (Elazar et al., 2023). Researchers have raised serious concerns about these issues (Jacovi et al., 2023; Sainz et al., 2023). For arithmetic reasoning specifically, Zhang et al. (2024) recently found evidence that the widely used GSM8k benchmark (Cobbe et al., 2021) is contaminated, and presented evaluations on a new dataset which is not publicly released. MathGAP mitigates data contamination by creating new, more complex synthetic test sets.

Generalization on reasoning tasks. Generalization to data from a *different* distribution (i.e., OOD data) is a core problem in machine learning and has been studied extensively also in the context of LLM reasoning (Schwarzschild et al., 2021; Razeghi et al., 2022; An et al., 2023; Kudo et al., 2023; Liu et al., 2023; Saparov & He, 2023; Zhang et al., 2023; Thomm et al., 2024). While LLMs' performance on OOD data can be improved by various techniques (Anil et al., 2022; Borazjanizadeh & Piantadosi, 2024; Hu et al., 2024), generalization to problems of arbitrary complexity under a set of known inference rules is still an open problem. For instance, Dziri et al. (2023) demonstrate that the accuracy of transformer-based LLMs on compositional generalization rapidly approaches zero as the complexity of the problem increases—regardless of in-context guidance. Saparov et al. (2023) focus on theorem proving for logical reasoning, presenting a systematic analysis on generalization ability of LLMs with respect to proof size and diversity of inference rules. Our paper contributes to this literature by providing an evaluation framework to systematically study OOD generalization in regards to arithmetic proof complexity.

Evaluation on MWP benchmarks. It is common to treat MWPs as a testbed for studying reasoning abilities of LLMs (Cobbe et al., 2021; Patel et al., 2021; Fu et al., 2023; Shakarian et al., 2023; Stolfo et al., 2023; Zong & Krishnamachari, 2023); such problems are useful because solving them requires several distinct skills, yet, they remain conceptually simple (Stern, 1993). However, we are not aware of many studies that investigate generalization in regards to problem complexity in this domain. Our study presented in §5 relates to Hase et al. (2024), who fine-tune LLMs on easy problems and evaluate them on harder ones at inference. Their complexity metrics focus on problem length, but do not capture *how* a longer reasoning problem gets more complex (i.e., the shape of the proof tree). Moreover, they evaluate on GSM8k, which is contaminated. Mishra et al. (2024) proposed a method that uses GPT-4 to generate new synthetic problems from symbolic specifications. Their approach is scalable, but using an LLM to generate problems may (i) introduce bias (Boguraev et al., 2024; Opedal et al., 2024) and (ii) produce problems that are unfaithful to their specifications. Mirzadeh et al. (2024) recently performed evaluations on a version of GSM8k in which the variable names and numbers have been altered, and found that LLMs are surprisingly sensitive to such substitutions. MathGAP is more general; while it allows for such substitutions as well, our main goal is to vary the proof structures.

3 A Formal Treatment of Math Word Problems

We aim to generate new problems of arbitrary complexity, so we require a formalism for characterizing complexity in a precise manner. First, we explain how the semantics expressed in math word problems can be written as sequences of logical forms (§3.1). We then show how to apply inference rules on these logical forms to deduce new ones (§3.2). This leads to a view of problem solutions as proof trees, the structure of which can be used to characterize the reasoning required to solve the problem.

3.1 Math word problems as logical forms

Each math word problem is represented as a sequence of logical forms under the formalism from Opedal et al. (2023; 2024). A logical form is a statement about a truth in the world that is being described by the problem, representing some arithmetic relationship between the number of entities possessed by one or several agents. Such statements can refer to the number of entities an agent has, how many more entities an agent has relative to another, the action of an agent giving their entities to another agent, etc. Formally, a logical form represents the semantics of a single sentence and

Table 1: Examples of logical forms. A logical form consists of a predicate and a set of properties which is specific to that predicate, given here in separate columns. Each logical form induces a set of natural language sentences which preserve its semantics.

Logical Form		**Example Templates**	**Example Sentences**
Predicate (abbr.)	**Properties**		
container (cont)	(agent=a, quantity=q, entity=e, attribute=k, unit=u)	*[a] has [q] [u]s of [k] [e]s.*	*Alice has 5 kilograms of red apples.*
		[a] owns [q] [u]s of [k] [e]s.	*Alice owns 5 kilograms of red apples.*
comparison (comp)	(agentA=a, agentB=b, quantity=q, entity=e)	*[b] has [q] fewer [e]s than [a].*	*Bob has 3 fewer apples than Alice.*
		[a] has [q] more [e]s than [b].	*Alice has 3 more apples than Bob.*
transfer	(receiver_agent=b, sender_agent=a, quantity=q, entity=e)	*[a] gave [b] [q] [e]s.*	*Alice gave Bob 3 apples.*
		[b] got [q] more [e]s from [a].	*Bob got 3 more apples from Alice.*
partwhole	(whole_agent=$\wedge_{i=1}^{n} a_i$, part_agent1=a_1, ... part_agentn=a_n, whole_entity=f, part_entity=e)	*[a_1], ..., and [a_n] combine the [f]s that they have.*	*Alice and Bob combine the fruits that they have.*
comp-eq	(agentA=a, agentB=b, agentC=c, agentD=d, entity=e)	*The number of [e]s that [c] has more than [d] is equal to the difference between the number of [e]s that [a] and [b] have.*	*The number of apples that Charlie has more than David is equal to the difference between the number of apples that Alice and Bob have.*

consists of a **predicate** that takes a set of **properties** as arguments. Every logical form has an **agent** property, an **entity** property, and a **quantity** property. Other, optional, properties include **unit** and **attribute**. We define a **world model** as a sequence of logical forms that align with the sentences of a problem text, thus, representing the semantics of the entire problem. See Fig. 5 for the world model representation of the problem from Fig. 1.

For a given predicate and set of properties, we write `predicate(property1=x, property2=y, ...)` to express a logical form. Property names are, however, often omitted for brevity, as in `predicate(x, y, ...)`. There are two types of predicates: (i) those that represent ownership over some entity by an agent, called `container` (abbreviated `cont`), and (ii) those that represent arithmetic relationships between two sets of properties. The relationship predicates correspond to arithmetic concepts and are based on a taxonomy from the learning sciences (Riley et al., 1983); we use `transfer`, `comparison` (abbreviated `comp`), `partwhole` and `comp-eq`.[2] Table 1 gives concrete example sentences for logical forms with each of these predicates and App. A provides further intuition.

The logical forms in the world model of a problem are separated into the **body** and **question**. The question is a single logical form that appears at the end of the world model. It is characterized by having a placeholder variable instead of an explicit quantity, representing the correct answer to the problem. For example, the interrogative sentence *"How many apples does Alice have?"* is represented by the question `cont(Alice, q, apples)`, where `q` is a placeholder variable. The body consists of all logical forms in the world model which are not questions. The **answer** to a problem is the logical form declared by the question with the correct numerical quantity substituted into its placeholder variable, although we will sometimes refer to only the correct numerical quantity as the answer as well.

3.2 Chain-of-thought solutions as proof trees

Solving math word problems is a form of deductive reasoning, where the correct answer follows necessarily from what is stated in the text through a combination of rules of arithmetic and world knowledge. We use such derivations, called proofs, to characterize the structure of problems. The proofs in our setup are analogous to those found in other deductive systems, e.g., of first-order logic.

[2]The transfer and part-whole concepts are usually called *change* and *combine*, respectively, in learning science literature (Nesher et al., 1982). The `comp-eq` predicate was introduced in this paper in order to construct nonlinear problems of arbitrary length.

Table 2: Inference rule templates (**left**) with corresponding examples of rule applications in natural language (**right**). The variables a, b, c, d refer to agents, q, $\mathtt{q}_1$, $\mathtt{q}_2$ refer to quantities, and e, f refer to entities. Attribute and unit properties are excluded here, but analogous rules exist in which they are present. All inference rules are commutative except the `transfer` rule; see footnote 3. Rule applications of an inference rule with premises $L_1, L_2, \ldots, L_N$ and consequent L are written as $L'_1.L'_2.\ \ldots\ .L'_N. \vdash L'.$, where L' is a natural language expression for the logical form L. Axioms are formed from the facts stated in the problem, as illustrated in Fig. 1.

Inference Rules	**Example Sentences**
$\dfrac{\mathtt{cont(a,\ q_1,\ e)}\quad \mathtt{comp(b,\ a,\ q_2,\ e)}}{\mathtt{cont(b,\ q_1\ +\ q_2,\ e)}}$	*Alice has 3 apples. Bob has 2 more apples than Alice. ⊢ Bob has 5 apples.*
$\dfrac{\mathtt{cont(a,\ q_1,\ e)}\quad \mathtt{transfer(a,\ b,\ q_2,\ e)}}{\mathtt{cont(a,\ q_1\ +\ q_2,\ e)}}$	*Alice has 3 apples. Bob gave 2 apples to Alice. ⊢ Alice has 5 apples.*
$\dfrac{\mathtt{cont(a,\ q_1,\ e)}\quad \mathtt{cont(b,\ q_2,\ e)}}{\mathtt{comp(b,\ a,\ q_2\ -\ q_1,\ e)}}$	*Alice has 3 apples. Bob has 5 apples. ⊢ Bob has 2 more apples than Alice.*
$\dfrac{\mathtt{cont(a_1,\ q_1,\ e)}\ldots\mathtt{cont(a}_n\mathtt{,\ q}_n\mathtt{,\ e)}\quad \mathtt{partwhole(}\wedge_{i=1}^{n}\mathtt{a}_i\mathtt{,\ a_1,\ \ldots,\ a}_n\mathtt{,\ f,\ e)}}{\mathtt{cont(}\wedge_{i=1}^{n}\mathtt{a}_i\mathtt{,\ }\sum_{i=1}^{n}\mathtt{q}_i\mathtt{,\ f)}}$	*Alice has 3 apples. Bob has 5 apples. Alice and Bob combine their fruits. ⊢ Alice and Bob have 8 fruits.*
$\dfrac{\mathtt{cont(a,\ q_1,\ e)}\quad \mathtt{comp(d,\ c,\ q_2,\ e)}\quad \mathtt{comp\text{-}eq(b,\ a,\ d,\ c)}}{\mathtt{cont(b,\ q_1\ +\ q_2,\ e)}}$	*Alice has 7 apples. David has 2 more apples than Charlie. The number of apples that Bob has more than Alice is the same as the difference between the number of apples that David and Charlie have. ⊢ Bob has 9 apples.*

We want to formally reason over the logical forms introduced in §3.1. To that end we introduce **inference rules**, which are used to prove new logical forms from previously-known ones. A **proof step**, written as

$$\frac{L_1 \qquad L_2 \qquad \cdots \qquad L_N}{L},$$

is an instance of an inference rule where the **conclusion** logical form L is deduced from **premises** $L_1, L_2, \ldots, L_N$. An **axiom** is a proof step without premises. The axioms of a math word problem are the logical forms in the body of its world model, i.e., all logical forms excluding the question. Table 2 lists the inference rules that we use, along with examples of how they may be expressed in natural language.[3] For instance, consider the two logical forms `cont(Isabella, 17, apple)` and `comp(Lucy, Isabella, 10, apple)` from Fig. 1, representing the facts that Isabella has 17 apples and Lucy has 10 more apples than Isabella, respectively. The proof step

$$\frac{\mathtt{cont(Isabella,\ 17,\ apple)}\quad \mathtt{comp(Lucy,\ Isabella,\ 10,\ apple)}}{\mathtt{cont(Lucy,\ 17\ +\ 10,\ apple)}},$$

lets us deduce the logical form `cont(Lucy, 27, apple)`, i.e., that Lucy has 27 apples.

A math word problem can be solved by applying inference rules until the answer to the problem has been proved. Formally, a **proof tree** is a rooted ordered tree where each node is labeled with a logical form L, which is the conclusion of a proof step with premises $L_1, \ldots, L_N$, matching the labels of the child nodes in the same order. Leaf nodes are thus labeled with axioms. A **proof** of a *particular* problem is a proof tree where the labels on the leaf nodes are logical forms in the body of the problem, and the label at the root is the answer to the problem. The tree shown in Fig. 1 illustrates a proof of the problem given in the same figure.

Proof complexity. We seek a systematic way to analyze whether LLMs can generalize from simple to more complex proofs. The definition of proof given above gives rise to several ways of characterizing reasoning, out of which we consider four: linearity, depth, width, and ordering.

[3]The list is not exhaustive as additional variations on the rules exist. In particular, the logical forms may contain attributes and units (such as the `cont` examples in Table 1), and the premises can be given in any order for all inference rules (i.e., they are commutative) except the one using `transfer`. The `transfer` rule is non-commutative since it involves a notion of time. For instance, "Alice (now) has 5 apples. Alice ate 2 apples." does not imply the same conclusion as "Alice ate 2 apples. Alice (now) has 5 apples." Moreover, note that the third inference rule listed in Table 2 is commutative, since swapping the two premises yields the conclusion $\mathtt{comp(a,\ b,\ q_1 - q_2,\ e)}$, which is equivalent to $\mathtt{comp(b,\ a,\ q_2 - q_1,\ e)}$.

We say that a proof tree is **linear** if every one of its inference rules takes at most one premise that is not an axiom, and **nonlinear** otherwise. In other words, every step in the solution to a linear problem only uses at most one fact not directly present in the problem. The proof tree given in Fig. 1 is nonlinear because the proof step that deduces the answer uses two premises that are not themselves axioms. The **depth** of a proof tree is defined as its height, i.e., the maximum path length from the root node to any leaf node. Next, the **width** of a proof tree is the number of leaf nodes, or axioms, that it contains. The proof tree in Fig. 1 thus has depth 2 and width 5. In words, this means that the answer is two proof steps "away" from any of the axioms and that the problem has five (declarative) sentences. Lastly, we define the **canonical ordering** to be the ordering of leaf nodes as visited left-to-right. Note that the sentences in the natural language annotation from Fig. 1 follow the canonical ordering of the proof tree. Informally, this is the easy way of ordering the sentences in the problem since it matches the ordering of the proof steps. However, there exist multiple alternative orderings in general. In particular, given a proof tree that only contains commutative inference rules (see footnote 3), all orderings of the leaf nodes are valid. For instance, swapping the ordering of the first two leaf nodes in the tree in Fig. 1 yields a problem with the same proof.

4 Evaluation Framework

Equipped with the formalism explained in §3, we propose an evaluation framework that tests LLMs on MWPs of arbitrary proof complexity, called MathGAP. In §4.1, we explain the generation method behind MathGAP. We then outline how this method can be used for OOD evaluation in §4.2.

4.1 Generation Method

We propose a method for synthetic generation of problems and their CoT solution explanations. It consists of three high-level steps: (i) sample a proof tree, (ii) map the logical forms at the leaf nodes into a natural language problem, and (iii) map the proof steps into a CoT solution.[4] Fig. 1 gives an illustration of these steps; note that the agent properties are abbreviated for brevity.

In detail, the procedure is as follows: We first sample a logical form at the root of the proof tree. We then sample a specific inference rule as well as its premises. We recursively repeat this procedure for each premise until a predetermined stopping criterion has been reached, resulting in a proof tree. The stopping criterion is user-specified; it could be, e.g., when the tree has reached a certain depth or width. The categorical properties for the logical forms (i.e., agent, entity, attribute, and unit) are sampled uniformly at random from some vocabularies[5] The quantities are sampled uniformly at random from a user-specified range; in our study presented in §5 we use $2-20$.

We then traverse the leaves of the proof tree in some order and convert each into a sentence via a natural language template, forming the text corresponding to the body of the problem. The default order follows the proof tree's canonical ordering, but other orders are possible as well. The sentences are sampled uniformly at random from a predefined set of templates; some example templates are given in Table 1. The question is converted from the logical form at the root of the proof tree and is sampled from a set of interrogative sentences.

In the final step, we generate the CoT annotation for the problem. We first obtain a sequential ordering of the CoT proof steps by visiting the nodes in the proof tree with a post-order traversal.[6] We then convert each proof step into a sentence via a natural language template. The templates used for the axioms are identical to those used in the body of the problem text. In other words, the proof steps corresponding to axioms simply repeat the relevant part of the problem text. For the other proof steps, we have templates that are designed to explain the computation step that is performed. Fig. 1 illustrates the mapping between nodes in the proof tree and sentences in the CoT annotation. For

[4] Jin et al. (2023) propose a related method for generating causal reasoning problems.

[5] For our experiments presented in §5, we use a handwritten list of 52 English-language names for most problems, with the exception of the larger nonlinear problems for which we use a larger list of 4,945 predominantly English-language names from https://github.com/dominictarr/random-name. For the entities we use a handwritten word list with 51 items.

[6] Note that the choice of traversing the nodes in post order corresponds to applying each inference rule as soon as all of its premises are available. In addition, post-order traversal visits the axioms in the canonical ordering. Alternative orders of tree traversal would violate at least one of these constraints.

instance, the third sentence in Fig. 1 explains the conclusion `cont(Lucy, 27, apple)`, which is deduced from the two leftmost nodes in the tree: "So Lucy has $17 + 10 = 27$ apples".

Our pipeline is similar to the one proposed by Opedal et al. (2024), except that we use a recursive procedure to generate each proof step starting at the root, instead of a sequential one. This is a more general approach that also enables generation of nonlinear proofs.

4.2 MathGAP: Mathematical Generalization on Arithmetic Proofs

With the generation method just described, we can generate new synthetic problems where we control for the structure of the proof and its complexity. MathGAP can be used for various forms of systematic studies on arithmetic reasoning; here, we focus on OOD generalization. Specifically, we can select a family of problems of interest (e.g., linear problems under `comp` inference rules) and generate a train-test split of problems for which the problems in the test set are more complex than those in the training set (e.g., containing problems with deeper proofs).

The generation method can be flexibly adapted to fit specific studies. One can add new logical forms and inference rules, restrict the choice of existing ones during sampling, or create logical forms for irrelevant sentences (Shi et al., 2023). One can also vary the text templates for the logical forms and inference rules, the vocabularies for agents, entities, attributes, and units, and the range of numbers from which the quantities are sampled. In addition, one may increase linguistic diversity by incorporating a last step in which a language model paraphrases the templated texts, or, alternatively, use a procedure like the one proposed by Mishra et al. (2024). In our study, we do not perform such paraphrasing, since we want to guarantee that the sentences are faithful to the proof trees from which they are generated. However, future studies using MathGAP may choose to prioritize differently.

5 Experiments

We apply MathGAP to study how LLMs generalize from proof demonstrations of simple problems in context to solve problems of higher complexity at inference. We test OOD generalization through several sets of experiments: depth generalization for linear problems (§5.1), width generalization for linear problems (§5.2), depth generalization for nonlinear problems (§5.3), and generalization to permutations (§5.4). We are also interested in how the choice of the distribution of problems provided in context affects performance, which is often not obvious (Min et al., 2022; Turpin et al., 2023).

Experimental setup. For each set of experiments, the general experimental setup is as follows: We generate five test sets of different degrees of complexity with 400 problems in each. We then generate model predictions for the problems in these test sets under four different prompts: **(i)** A *zero-shot* baseline prompt, which contains only the test problem; **(ii)** A prompt of *primitive* examples, in which all problems contain only one proof step of the same inference rule that is used in the test problem; **(iii)** A prompt of examples within a *range* of complexities, all of which are simpler than the test problem; **(iv)** An *in-distribution* baseline prompt, containing examples that are of the same complexity as the test problem.

Note that prompts (i) through (iii) evaluate in-context OOD generalization, since the test examples are from a distribution that is different from that of the in-context examples. For prompts (ii) through (iv) we generate a new set of in-context examples for every test problem. This is done in order to avoid bias in the results towards any particular set of examples. We set the number of examples to 12, except for the experiments on nonlinear problems for which we use 5.[7] Importantly, the in-context problems are provided together with their CoT solution annotations, as generated through the MathGAP pipeline. They are written out using the pattern "*Q:* `{problem}\n`*A:* `{CoT solution}\n`". For prompt (iii), we make sure that there is at least one problem of every complexity in the range. This prompt can be viewed as a form of in-context curriculum learning (Bengio et al., 2009),[8] except that we randomize the ordering of the examples rather than arranging them in increasing order of complexity.

[7] We use fewer in-context examples for nonlinear problems since the number of sentences in a nonlinear problem increases exponentially with depth. The number 12 was deemed large enough so that additional examples would have a negligible positive impact on performance; see, e.g., Agarwal et al. (2024, Fig. 7). Moreover, in preliminary experiments we found that the number of in-context examples had little effect on performance.

[8] Related approaches have been explored in previous work (Li et al., 2022; Liu et al., 2024b).

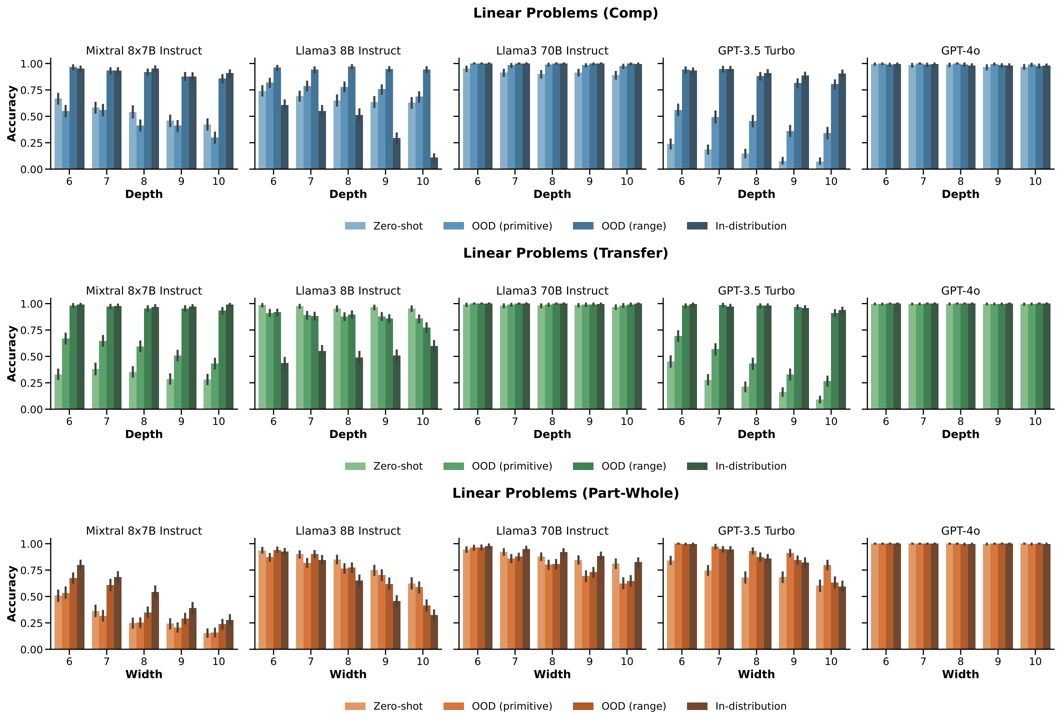

Figure 2: Answer accuracies for generalization to increasing depth and width for linear problems, across models and in-context distributions. Depth is increased using inference rules involving `comp` and `transfer`. Width is increased using `partwhole`.

This is done since in-context orderings can have a large effect on performance (Lu et al., 2022), and we do not want our results to overfit to any particular ordering.

Responses are generated using greedy decoding and a maximum length of 4,096 tokens. Model predictions are obtained by extracting the last number occurring in the model's generated output. We report answer accuracies and compute 95% confidence intervals using bootstrap sampling (Efron, 1992). We evaluate the following models on all test sets: Mixtral-8x7B (Jiang et al., 2024a), Llama3 with 8B and 70B parameters (Llama Team, 2024), GPT-3.5 Turbo and GPT-4o (OpenAI, 2024).

5.1 Linear Depth Generalization

Problem sets. We consider linear problems using both `comp` and `transfer`. The five test sets consist of problems with depths between $6-10$. As such, the examples provided in the range of setting (iii) described above have depths between $1-5$. We give an example of a problem with depth 6 in App. B.1.

Results. We illustrate the results in the top and middle rows of Fig. 2. As expected, we observe a decreasing trend in performance as depth increases, for all in-context distributions. Curiously, GPT-3.5-Turbo is the only model for which adding in-context examples consistently leads to improvements. For the majority of cases, performance tends to be larger when providing a range of examples of varying complexity as compared to only primitive examples. The ranged OOD cases often show comparable performance to that of the in-distribution context, suggesting that these models benefit from demonstrations of how problems of varying complexity are solved. Llama3-8B is an outlier in that its solving accuracy for in-distribution context is (substantially) lower as compared to other prompts. When inspecting the model outputs, we see that the model often fails to answer the inference question, recognizing instead that it has been provided a sequence of in-context examples. This could potentially be a consequence of how Llama3 is trained, using smaller context windows for the initial stages of pre-training (Llama Team, 2024). This is remedied in Llama3 70B, which shows near perfect performance across all test sets. The difference in performance between the two Llama3 models can most likely be explained by scale, since they are trained using the same procedure and data (Llama Team, 2024).

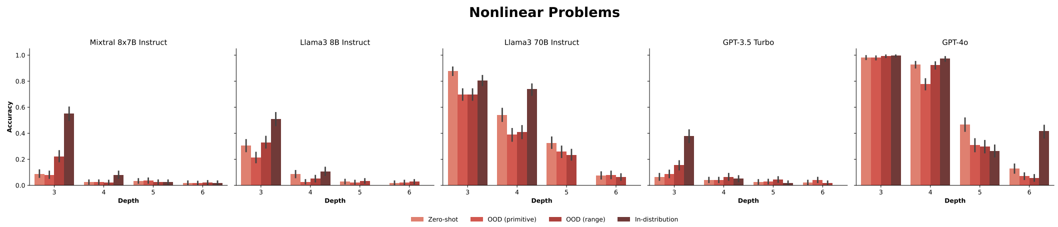

Figure 3: Answer accuracies for generalization to increasing depth and width for nonlinear problems, across models and in-context distributions. Depth is increased using inference rules involving `comp` and `comp-eq`. The in-distribution contexts were too large to fit in the prompts for the Llama3 models and GPT-3.5 Turbo at some of the higher depths. App. C shows additional results on o1-preview.

5.2 Linear Width Generalization

Problem sets. To test generalization to greater proof width, we use `partwhole` problems with a fixed depth of 1. The five test sets have widths between $6-10$ and the width of the problems in setting (iii) described above have widths between $1-5$.

Results. The results are shown in the bottom row of Fig. 2. For GPT-3.5 and Mixtral-8x7B, we observe that providing in-context examples improves accuracy over zero-shot in most cases. As for linear depth generalization, it seems that having a range of OOD examples of varying complexity is superior to only including primitive examples. Llama3-8B exhibits worse performance with in-distribution context here as well. In general, for the Llama models, adding OOD in-context examples usually leads to lower performance over zero-shot. Apart from GPT-3.5-Turbo, the performance is generally lower as compared to a deep problem with the same number of sentences (§5.1), often with a sharper decreasing trend. Such is the case even for Llama3 70B, which performed almost perfectly in the linear depth setting. This would suggest that it is more difficult to generalize in terms of width than in terms of depth. Again, GPT-4o demonstrates near perfect performance across all test sets.

5.3 Nonlinear Depth Generalization

Problem sets. We generate test datasets containing nonlinear problems with `comp` and `comp-eq`, having the same form as the problem in Fig. 1. Another example is shown in App. B.2. We generate a test set for each depth between $3-6$. Thus, the ranged OOD context setting contains examples of depths between $1-2$.

Results. The results are shown in Fig. 3. All models exhibit a performance trend that tends to zero as depth increases, albeit at different rates. For Mixtral-8x7B, Llama3-8B, and GPT-3.5 Turbo, the performance decreases rapidly across all context modes. These models benefit from in-distribution contexts for low depths, and in some cases, from a range of OOD examples. Llama3-70B and GPT-4o are more robust to this type of OOD distribution shift, but for the deepest problems, their performance tends to zero as well. Moreover, it is worth noting that, for many cases, providing in-context examples does not significantly improve performance over zero-shot. On the other hand, for lower depths, it is usually beneficial to provide in-distribution contexts. We perform an additional analysis on o1-preview in App. C.

Comparison to linear problems. Nonlinear problems are more difficult than the linear ones, even when controlling for the number of axioms given in the problem (i.e., its width). To see this, note that a nonlinear problem with depth 3 has 10 axioms. This is the same number of axioms as the linear comparison and transfer problems (§5.1) with depth 9 and the linear part-whole problems (§5.2) with width 10. Comparing the results across those problem sets, we observe considerably lower accuracies for the nonlinear problems in most cases. We believe there are two main explanations for this: (i) Solving nonlinear problems requires keeping intermediate conclusions in memory while proving other intermediate conclusions, before being able to use them for further proof steps. This is in contrast to linear problems, in which intermediate conclusions are used immediately in a new proof step. (ii) Nonlinear problems use the `comp-eq` rule, which is most likely less frequent in the training set. Inspecting the model outputs we find support for both of these explanations—errors show up both when a previously deduced conclusion is needed as well as when a `comp-eq` is required in a proof step.

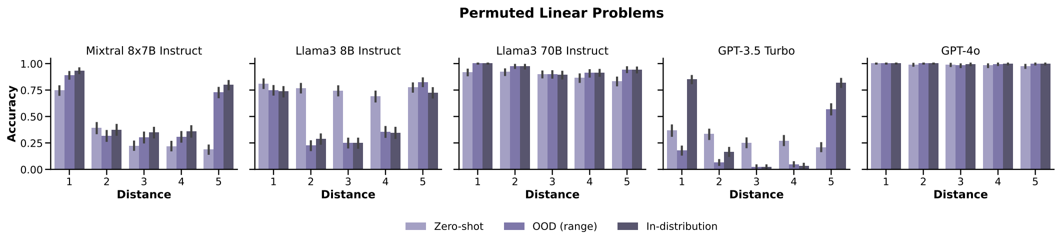

Figure 4: Answer accuracies for generalization to permutations, across models and in-context distributions. We measure complexity as the distance of the movement of a sentence to the beginning of the problem, as compared to its position in the canonical ordering of the problem.

5.4 Order Generalization

Problem sets. LLMs are known to be sensitive to changes in the order of axioms in logical and mathematical reasoning problems (Chen et al., 2024; Eisape et al., 2024). Here, we present a systematic analysis of their sensitivity to a particular form of order permutation. We consider linear, left-leaning proof trees using `comp` that have depth 5, and generate problems that deviate from the canonical, left-to-right order of the leaf nodes. In particular, we pick one of the leaf nodes to visit first, and then visit the remaining leaf nodes in left-to-right order. This has the effect of moving one of the sentences in the canonically-ordered problem text to the beginning of the problem. We characterize complexity as the distance of the movement, with the hypothesis that a greater movement from a canonical ordering constitutes a more difficult problem. We create five test sets with movement distances between $1-5$. We do not include a primitive OOD in-context distribution since the notion of a primitive problem is ill-defined in this case. The range prompt includes movements of all distances between $1-5$ except the one present in the test set.

Results. The results are shown in Fig. 4. First, note that all models show lower performance on perturbed problems as compared to non-perturbed ones (compare to Fig. 2). Second, there is a nonlinear relationship between accuracy and distance of the movement, in contrast to the monotonically decreasing relation we expected. More specifically, the performance is the highest for the problems where the sentence is moved from near the beginning or the end of the problem.[9] In addition, including in-context examples gives a larger boost in performance for short and long movements, as compared to medium-length ones. Inspecting the model outputs, we observe that the models often make logical mistakes when the sentence that has been moved is needed for the subsequent proof steps. These kinds of mistakes appear to be more common than arithmetic errors.

6 Conclusion

In this paper, we introduced MathGAP, a framework for evaluating LLMs on math word problems of arbitrary arithmetic proof complexity. MathGAP can flexibly generate synthetic arithmetic word problems with controllable proof tree characteristics. Importantly, this enables studies of OOD generalization that avoid issues of data contamination, since a train-test split where the distributions are different can be created programmatically. In particular, the test set can be made arbitrarily more complex than the training set. We apply MathGAP to study whether LLMs can learn from simple examples in context to solve more complex problems in a test set. All LLMs tested show a decrease in performance as proof complexity increases through depth and/or width. We also find that LLMs are sensitive to the order in which sentences are presented in a particular manner: A problem is harder if a sentence is moved from the middle of the problem, as compared to the beginning or end. We find no clear relationship between the distribution of in-context examples and performance; zero-shot or OOD contexts sometimes yield performance higher than or comparable to in-distribution contexts. Finally, we demonstrate that we can use MathGAP to construct problems that are challenging even for state-of-the-art reasoning models like OpenAI o1—see App. C. App. D discusses limitations of the present work.

[9]This is consistent with findings on retrieval-augmented generation, for which models are better at using information that occurs near the beginning or end of the prompt (Liu et al., 2024a).

REFERENCES

Rishabh Agarwal, Avi Singh, Lei M. Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, John D. Co-Reyes, Eric Chu, Feryal Behbahani, Aleksandra Faust, and Hugo Larochelle. Many-shot in-context learning, 2024. URL https://arxiv.org/abs/2404.11018.

Shengnan An, Zeqi Lin, Qiang Fu, Bei Chen, Nanning Zheng, Jian-Guang Lou, and Dongmei Zhang. How do in-context examples affect compositional generalization? In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11027–11052, Toronto, Canada, July 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.acl-long.618.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 38546–38556. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/fb7451e43f9c1c35b774bcfad7a5714b-Paper-Conference.pdf.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 41–48, New York, NY, USA, 2009. Association for Computing Machinery. URL https://doi.org/10.1145/1553374.1553380.

Sasha Boguraev, Ben Lipkin, Leonie Weissweiler, and Kyle Mahowald. Models can and should embrace the communicative nature of human-generated math, 2024. URL https://arxiv.org/abs/2409.17005.

Nasim Borazjanizadeh and Steven T. Piantadosi. Reliable reasoning beyond natural language, 2024. URL https://arxiv.org/abs/2407.11373.

Xinyun Chen, Ryan Andrew Chi, Xuezhi Wang, and Denny Zhou. Premise order matters in reasoning with large language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=4zAHgkiCQg.

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

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. Investigating data contamination in modern benchmarks for large language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8706–8719, Mexico City, Mexico, June 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.naacl-long.482.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1286–1305, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.emnlp-main.98.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang (Lorraine) Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 70293–70332. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/deb3c28192f979302c157cb653c15e90-Paper-Conference.pdf.

Bradley Efron. Bootstrap methods: Another look at the jackknife. In *Breakthroughs in Statistics: Methodology and Distribution*, pp. 569–593. Springer, 1992. URL https://link.springer.com/chapter/10.1007/978-1-4612-4380-9_41.

Tiwalayo Eisape, Michael Tessler, Ishita Dasgupta, Fei Sha, Sjoerd Steenkiste, and Tal Linzen. A systematic comparison of syllogistic reasoning in humans and language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8425–8444, Mexico City, Mexico, June 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.naacl-long.466.

Yanai Elazar, Akshita Bhagia, Ian Helgi Magnusson, Abhilasha Ravichander, Dustin Schwenk, Alane Suhr, Evan Pete Walsh, Dirk Groeneveld, Luca Soldaini, Sameer Singh, Hanna Hajishirzi, Noah A. Smith, and Jesse Dodge. What's in my big data? In *The Twelfth International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2310.20707.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=yf1icZHC-l9.

Peter Hase, Mohit Bansal, Peter Clark, and Sarah Wiegreffe. The unreasonable effectiveness of easy training data for hard tasks. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7002–7024, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.378.

Yi Hu, Xiaojuan Tang, Haotong Yang, and Muhan Zhang. Case-based or rule-based: How do transformers do the math? In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 19438–19474. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/hu24n.html.

Alon Jacovi, Avi Caciularu, Omer Goldman, and Yoav Goldberg. Stop uploading test data in plain text: Practical strategies for mitigating data contamination by evaluation benchmarks. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5075–5084, Singapore, December 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.emnlp-main.308.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024a. URL https://arxiv.org/abs/2401.04088.

Bowen Jiang, Yangxinyu Xie, Zhuoqun Hao, Xiaomeng Wang, Tanwi Mallick, Weijie J Su, Camillo Jose Taylor, and Dan Roth. A peek into token bias: Large language models are not yet genuine reasoners. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4722–4756, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. URL https://aclanthology.org/2024.emnlp-main.272.

Zhijing Jin, Yuen Chen, Felix Leeb, Luigi Gresele, Ojasv Kamal, Zhiheng LYU, Kevin Blin, Fernando Gonzalez Adauto, Max Kleiman-Weiner, Mrinmaya Sachan, and Bernhard Schölkopf. CLadder: A benchmark to assess causal reasoning capabilities of language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=e2wtjx0Yqu.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking benchmarking in NLP. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4110–4124, Online, June 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.naacl-main.324.

Keito Kudo, Yoichi Aoki, Tatsuki Kuribayashi, Ana Brassard, Masashi Yoshikawa, Keisuke Sakaguchi, and Kentaro Inui. Do deep neural networks capture compositionality in arithmetic reasoning? In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 1351–1362, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.eacl-main.98.

Changqun Li, Linlin Wang, Xin Lin, Gerard de Melo, and Liang He. Curriculum prompt learning with self-training for abstractive dialogue summarization. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 1096–1106, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.72.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=De4FYqjFueZ.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024a. URL https://aclanthology.org/2024.tacl-1.9.

Yinpeng Liu, Jiawei Liu, Xiang Shi, Qikai Cheng, Yong Huang, and Wei Lu. Let's learn step by step: Enhancing in-context learning ability with curriculum learning, 2024b. URL https://arxiv.org/abs/2402.10738.

Llama Team. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098, Dublin, Ireland, May 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.acl-long.556.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11048–11064, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.759.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models, 2024. URL https://arxiv.org/abs/2410.05229.

Shubhra Mishra, Gabriel Poesia, Belinda Mo, and Noah D. Goodman. Mathcamps: Fine-grained synthesis of mathematical problems from human curricula, 2024. URL https://arxiv.org/abs/2407.00900.

Pearla Nesher, James G. Greeno, and Mary S. Riley. The development of semantic categories for addition and subtraction. *Educational Studies in Mathematics*, 13:373–394, 1982. URL https://api.semanticscholar.org/CorpusID:120936397.

Andreas Opedal, Niklas Stoehr, Abulhair Saparov, and Mrinmaya Sachan. World models for math story problems. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 9088–9115, Toronto, Canada, July 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.findings-acl.579.

Andreas Opedal, Alessandro Stolfo, Haruki Shirakami, Ying Jiao, Ryan Cotterell, Bernhard Schölkopf, Abulhair Saparov, and Mrinmaya Sachan. Do language models exhibit the same cognitive biases in problem solving as human learners? In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024. URL https://arxiv.org/abs/2401.18070.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, Online, June 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.naacl-main.168.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. Impact of pretraining term frequencies on few-shot numerical reasoning. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 840–854, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.findings-emnlp.59.

Mary Riley, James Greeno, and Joan Heller. *Development of Children's Problem-Solving Ability in Arithmetic*, pp. 153–196. Learning Research and Development Center, University of Pittsburgh, 01 1983. URL https://eric.ed.gov/?id=ED252410.

Oscar Sainz, Jon Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10776–10787, Singapore, December 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.findings-emnlp.722.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=qFVVBzXxR2V.

Abulhair Saparov, Richard Yuanzhe Pang, Vishakh Padmakumar, Nitish Joshi, Mehran Kazemi, Najoung Kim, and He He. Testing the general deductive reasoning capacity of large language models using OOD examples. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=MCVfX7HgPO.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 6695–6706. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/3501672ebc68a5524629080e3ef60aef-Paper.pdf.

Paulo Shakarian, Abhinav Koyyalamudi, Noel Ngu, and Lakshmivihari Mareedu. An independent evaluation of chatgpt on mathematical word problems (MWP). In Andreas Martin, Hans-Georg Fill, Aurona Gerber, Knut Hinkelmann, Doug Lenat, Reinhard Stolle, and Frank van Harmelen (eds.), *Proceedings of the AAAI 2023 Spring Symposium on Challenges Requiring the Combination of Machine Learning and Knowledge Engineering (AAAI-MAKE 2023), Hyatt Regency, San Francisco Airport, California, USA, March 27-29, 2023*, volume 3433 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023. URL https://ceur-ws.org/Vol-3433/paper8.pdf.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Huai hsin Chi, Nathanael Scharli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, 2023. URL https://api.semanticscholar.org/CorpusID:256459776.

Elsbeth Stern. What makes certain arithmetic word problems involving the comparison of sets so difficult for children? *Journal of Educational Psychology*, 85:7–23, 1993. URL https://ethz.ch/content/dam/ethz/special-interest/gess/ifv/professur-lehr-und-lernforschung/publikationen-stern/A11_What_makes_certain_arithmetic_word_problems_involving_the_comparison_of_sets_so_hard_for_children_1993.pdf.

Alessandro Stolfo, Zhijing Jin, Kumar Shridhar, Bernhard Schölkopf, and Mrinmaya Sachan. A causal framework to quantify the robustness of mathematical reasoning with language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 545–561, Toronto, Canada, July 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.acl-long.32.

Jonathan Thomm, Aleksandar Terzic, Giacomo Camposampiero, Michael Hersche, Bernhard Schölkopf, and Abbas Rahimi. Limits of transformer language models on learning to compose algorithms. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2402.05785.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=bzs4uPLXvi.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Dylan Slack, Qin Lyu, Sean Hendryx, Russell Kaplan, Michele Lunati, and Summer Yue. A careful examination of large language model performance on grade school arithmetic, 2024. URL https://arxiv.org/abs/2405.00332.

Yi Zhang, Arturs Backurs, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, and Tal Wagner. Unveiling transformers with lego: a synthetic reasoning task, 2023. URL https://arxiv.org/abs/2206.04301.

Mingyu Zong and Bhaskar Krishnamachari. Solving math word problems concerning systems of equations with gpt-3. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(13): 15972–15979, Sep. 2023. URL https://ojs.aaai.org/index.php/AAAI/article/view/26896.

A Intuition on Logical Forms

A `cont` logical form expresses the fact that an agent owns a particular entity in some quantity (e.g., *"Alice has 5 apples"*). A `comp` logical form expresses how much more some agent owns of a particular entity than some other agent (e.g., *"Bob has 3 fewer apples than Alice"*). A `transfer` logical form expresses the transfer of a certain quantity of an entity from one agent to another (e.g.,

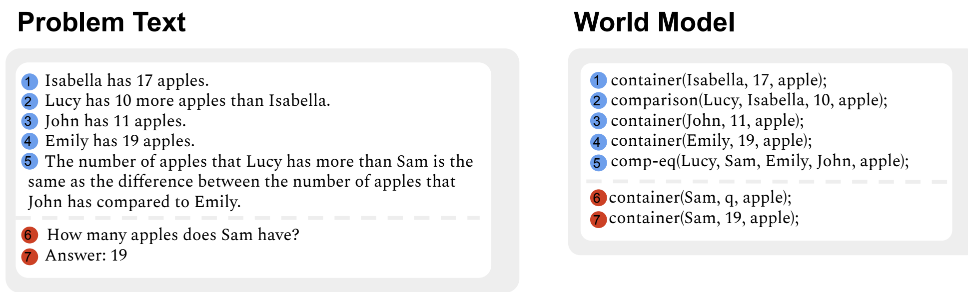

Figure 5: World model (**right**) of a math word problem (**left**). Each sentence in the problem text is represented by a logical form which consists of a predicate with property arguments. The logical forms in the body are used as axioms in the proof of the problem, as shown in Fig. 1. The sentences and logical forms labeled (1) through (5) are the body of the problem, (6) is the question and (7) is the answer.

"Alice gave Bob 3 apples"). A `partwhole` logical form expresses a superset relation between the combined entities of several agents (e.g., *Alice and Bob combine their fruits*). Finally, a `comp-eq` logical form expresses that the quantity of a particular `comp` between two agents is equal to that of a `comp` between two other agents (e.g., *"The difference between the number of apples that Alice and Bob have is the same as the difference between the number of apples that Charlie and David have"*).

Fig. 5 gives an example problem text along with its world model.

B Example Problems

In this section we provide example problems for each set of experiments (§§ 5.1 to 5.4).

B.1 Example linear comparison problem (depth 6)

Jackson has 16 red bottles of soap. Jackson has 10 more red bottles of soap than Abigail. Joseph has 18 more red bottles of soap than Abigail. Joseph has 14 fewer red bottles of soap than James. Michael has 2 more red bottles of soap than James. Ryan has 16 fewer red bottles of soap than Michael. Mia has 10 more red bottles of soap than Ryan. What is the number of red bottles of soap that Mia has?

B.2 Example nonlinear comparison problem (depth 3)

Ella has 11 yellow plates. Ella has 19 fewer yellow plates than Jacob. Evelyn has 16 yellow plates. Daniel has 10 yellow plates. The number of yellow plates that Emma has more than Jacob is the same as the difference between the number of yellow plates that Evelyn has compared to Daniel. Lucy has 2 yellow plates. Amelia has 6 more yellow plates than Lucy. Layla has 17 yellow plates. Layla has 13 fewer yellow plates than Sophia. The number of yellow plates that Emma has more than Henry is the same as the difference between the number of yellow plates that Amelia has compared to Sophia. What is the number of yellow plates that Henry has?

B.3 Example partwhole width problem (width 7)

Emily has 5 apples. Lily has 8 bananas. Abigail has 9 bananas. Benjamin has 11 grapes. Christopher has 20 apples. Mila has 16 grapes. Sophia has 11 watermelons. If everyone sums up the fruits that they have, how many fruits does everybody have in total?

B.4 Permuted problems

Canonically ordered problem text

Nicholas has 19 computers. Lucy has 6 fewer computers than Nicholas. Harper has 6 fewer computers than Lucy. John has 10 more computers than Harper. Abigail has 15 fewer computers

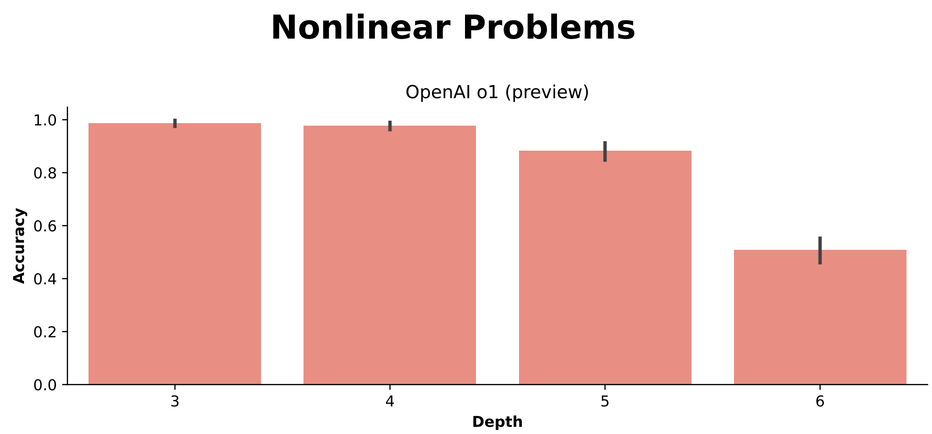

Figure 6: Answer accuracies for generalization to increasing depth and width for nonlinear problems for OpenAI o1-preview under zero-shot prompting.

than John. Abigail has 12 fewer computers than Logan. How many computers does Logan have in their collection?

PERMUTED PROBLEM TEXT (DISTANCE 1)

Lucy has 6 fewer computers than Nicholas. *Nicholas has 19 computers.* ***[Original Position]*** *Harper has 6 fewer computers than Lucy. John has 10 more computers than Harper. Abigail has 15 fewer computers than John. Abigail has 12 fewer computers than Logan. How many computers does Logan have in their collection?*

PERMUTED PROBLEM TEXT (DISTANCE 3)

John has 10 more computers than Harper. *Nicholas has 19 computers. Lucy has 6 fewer computers than Nicholas. Harper has 6 fewer computers than Lucy.* ***[Original Position]*** *Abigail has 15 fewer computers than John. Abigail has 12 fewer computers than Logan. How many computers does Logan have in their collection?*

PERMUTED PROBLEM TEXT (DISTANCE 5)

Abigail has 12 fewer computers than Logan. *Nicholas has 19 computers. Lucy has 6 fewer computers than Nicholas. Harper has 6 fewer computers than Lucy. John has 10 more computers than Harper. Abigail has 15 fewer computers than John.* ***[Original Position]*** *How many computers does Logan have in their collection?*

C ADDITIONAL ANALYSIS ON OPENAI O1

Fig. 6 shows additional results on the OpenAI o1-preview model, “o1-preview-2024-09-12”, evaluated on the nonlinear test sets. We only used zero-shot prompting due to the high costs of performing inference with o1-preview. Using a simple, straightforward prompt is also consistent with OpenAI’s current recommendations. We see that the o1 model shows superior performance to all models from the main text. However, as with the other models, the performance decreases as complexity increases.

Importantly, we note that performance can be improved by increasing the limit on the number of output tokens. The results presented in Fig. 6 used a token limit of 4,096, like the experiments in the main paper. We compared the effect of the token limit on an additional test set of 400 nonlinear problems with depth 7. With a token limit of 4,096, the model could only answer 0.25% of the problems correctly. However, with a token limit of 10,000, it answered 76.5% of the problems correctly. These results are noteworthy and suggests that the new inference technique used by the o1 model is highly effective in decomposing complex problems into smaller steps.

To test the model's limits, we performed a final evaluation on a subset of 100 of the depth 7 problems where the sentences in each problem were *randomly ordered*. Recall from §3.2 that permuting the sentences in random order is valid since all the inference rules used in these problems are commutative. In this experiment, we allowed 25,000 output tokens, which is the current recommendation by OpenAI. On this set, the answer accuracy was only 5%. Thus, we conclude that MathGAP is future-proof in the sense that it can generate problems on which current state-of-the-art models fail. We stress, however, that the aim of this paper is not to construct a challenging test set for state-of-the-art models; there are several other ways that MathGAP can be used to create problems that might be even more difficult.

D Limitations

Although we make sure to sample from a wide range of agent names, entities, attributes, and units, and include multiple templates for the predicates, future work may investigate how our findings generalize to distributions of problems with higher linguistic diversity. Thus, we do not consider the possibility of an agent token bias here (Jiang et al., 2024b). It is also beyond the scope of the present study to consider problem texts in languages other than English; however, it is straightforward to extend MathGAP to other languages. In addition, we cannot guarantee that the distributions of our generated problems are different from those of any potential internal data-generating mechanisms from Meta or OpenAI. However, we deem it to be unlikely that they use a similar formalism as we do here, and it gets increasingly unlikely that problems are similar as complexity increases.

While the more complex nonlinear problems are indeed challenging even for the most capable models, we note that the main purpose of our study was not to create a challenging evaluation set for state-of-the-art LLMs. If that is the goal, one could create harder problems by combining several inference rules (e.g., including the rate concept; Opedal et al., 2023), designing a wider variety of proof tree topologies, and presenting the axioms in arbitrary orderings. It is also possible to create more syntactically complex templates and use numbers on which it is more challenging for models to perform arithmetic (Razeghi et al., 2022).

Finally, it is hard to make a direct comparison between the difficulty of linear and nonlinear problems, since they use different inference rules. In particular, nonlinear proofs contain `comp-eq` predicates, which are not present in linear proofs.