Learning to Reason With Relational Abstractions

Andrew J. Nam*[1], Mengye Ren*[2], Chelsea Finn[1], James L. McClelland[1]
[1]Stanford University, [2]NYU

December 7, 2022

Abstract

Large language models have recently shown promising progress in mathematical reasoning when fine-tuned with human-generated sequences walking through a sequence of solution steps. However, the solution sequences are not formally structured and the resulting model-generated sequences may not reflect the kind of systematic reasoning we might expect an expert human to produce. In this paper, we study how to build stronger reasoning capability in language models using the idea of relational abstractions. We introduce new types of sequences that more explicitly provide an abstract characterization of the transitions through intermediate solution steps to the goal state. We find that models that are supplied with such sequences as prompts can solve tasks with a significantly higher accuracy, and models that are trained to produce such sequences solve problems better than those that are trained with previously used human-generated sequences and other baselines. Our work thus takes several steps toward elucidating and improving how language models perform on tasks requiring multi-step mathematical reasoning.

1 Introduction

Deep learning has had tremendous success in a wide range of domains, such as vision [He et al., 2016], language [Brown et al., 2020], and playing games at superhuman levels [Mnih et al., 2015, Silver et al., 2016, Vinyals et al., 2019]. Yet despite these accomplishments, reasoning yet appears to be limited, especially in areas of formal and mathematical reasoning [Saxton et al., 2019, Cobbe et al., 2021, Hendrycks et al., 2021], a puzzling phenomenon considering that many of the problems explored in previous research are ones that humans often successfully learn in primary and secondary education.

Recent findings in models of algorithmic reasoning suggest that neural networks, like humans, benefit from learning to solve mathematics through a chain of reasoning steps rather than attempting to produce the final output as a direct mapping from the problem prompt [Recchia, 2021, Nye et al., 2021, Hendrycks et al., 2021, Cobbe et al., 2021]. However, while various papers have explored how multi-step reasoning outperforms direct mapping, they often introduce and conduct analyses using new datasets each with different methods for how the individual steps are defined. This raises the question if and how the various formats of the reasoning steps affect learning differently. Unfortunately, this problem is difficult to address directly by simply comparing the performances between the datasets since they each contain questions of varying difficulty, potentially confounding the results. Moreover, some datasets use natural language [Cobbe et al., 2021, Hendrycks et al., 2021] which introduces further variance depending on the vocabulary and sentence structure, some avoid natural language altogether [Saxton et al., 2019, Lample and Charton, 2020], and some begin with natural language inputs but translate them into purely arithmetic operations while solving [Wang et al., 2017, Amini et al., 2019]. Thus, to understand how the solution structure impacts model learning, a common set of problems with varied solution approaches are needed.

In this work, we address this question and challenge by focusing on a natural language-based task, which requires integrating mathematical reasoning with world knowledge and coping with

*Equal Contribution.

the ambiguity of natural language, and a synthetic task that captures what we see as some of the central features of the mathematical reasoning. At their core, both tasks involve reasoning about how different entities relate to each other, and formulating appropriate arithmetic equations to perform the corresponding numerical computations. Thus, in both tasks, we can decompose each step of the solution into abstract relational reasoning and arithmetic expressions, which can then be used to recompose the solution sequence in different forms. We find that models, when prompted with the correct relational abstraction, can solve problems at a substantially higher accuracy, suggesting that relational abstraction is the more challenging component to single out in the problem solving process. Models that are trained with explicit relational abstractions also perform better than those that are not, which makes explicit relational abstraction a useful task for pre-training or fine-tuning.

We summarize our main contributions as follows:

- We propose to decompose the problem solving process into relational abstraction and arithmetic expression, which is amenable to large language models and is a middle-ground between neuro-symbolic, natural language-based, and arithmetic-only approaches.
- We find relational abstraction improves problem solving performance when supplied either during training or testing, making it a crucial component to study separately.
- We annotate the GSM-8K dataset [Cobbe et al., 2021] with relational abstractions and will release them as a supplementary dataset with the paper.
- We introduce a synthetic task paralleling aspects of natural relational reasoning and demonstrate the importance of engaging with the relational abstractions for solving this task.

2 Related Work

Models of mathematical reasoning. Although computational models of mathematical reasoning have been under research for over half a century [Bobrow, 1964], application of neural network models began much more recently using recurrent networks for sequence-to-sequence prediction [Wang et al., 2017]. Following the advent of the transformer model [Vaswani et al., 2017], Saxton et al. [2019] compared the efficacy of LSTMs, a relational network [Santoro et al., 2018], and transformers, and found that transformers well outperform the other architectures. However, this approach is limited in that the model is trained to produce only the final answer without any intermediate steps. Other models and datasets have been proposed to include intermediate equations and programs [Shi et al., 2015, Upadhyay and Chang, 2015, Amini et al., 2019, Miao et al., 2020]. Further advancements came with the availability of large language models [Brown et al., 2020, Thoppilan et al., 2022, Chowdhery et al., 2022, Lewkowycz et al., 2022] and datasets involving full step-by-step solutions in natural language [Ling et al., 2017, Hendrycks et al., 2021, Welleck et al., 2021, Cobbe et al., 2021, Drori et al., 2021].

Yet despite the optimism towards transformer-based models for mathematical reasoning [Lample and Charton, 2020], Hendrycks et al. [2021] argues that unlike in language benchmarks [Kaplan et al., 2020], simply scaling up the model size is an intractable strategy for solving mathematics problems, especially ones with of higher difficulty. Another strategy suggested by Cobbe et al. [2021] is the use of verifiers trained on model-generated responses to re-rank candidate sequences. Chain-of-thought prompting [Wei et al., 2022, Wang et al., 2022] circumvents fine-tuning altogether by providing examples at evaluation time as part of the context, and boasts comparable results to Cobbe et al. [2021] using just a few exemplars at test time, but such behavior only emerges on the largest language models with over 100B parameters. Compared to prior works, we propose using relational abstractions as an auxiliary outputs. For math word problems, we structure the relations in the form of natural language, and thereby leverage the existing knowledge built in large language models.

Learning with auxiliary language. The relational abstraction studied in this work can be cast as a form of auxiliary explanation to reach the goal. Beyond its role as the medium for step-by-step reasoning, language has been observed to assist neural networks in learning other tasks. Andreas et al. [2018] found that in rule-based image classification, string manipulation, and navigation tasks, generating and using language descriptions allowed the model to outperform those that learned to

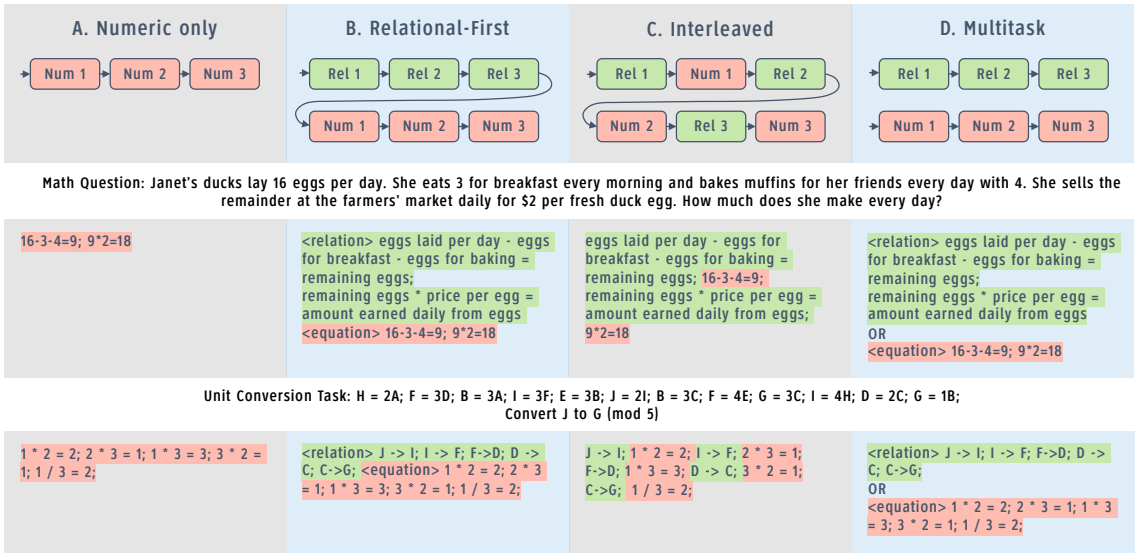

Figure 1: How can we incorporate structured relational reasoning in sequence-to-sequence modeling? Assuming that the mathematical reasoning process can be partitioned into the *abstract relational* and the *numeric* part, we explore four different possibilities: **A) Numeric only:** Only numeric steps are provided without any relational tokens; **B) Relational-first:** The abstract relational part are stated before the numeric; **C) Interleaved:** The abstract relational and numeric are interleaved; **D) Multitask:** Either output the abstract relational or the numeric but not both.

perform the tasks without language. Mu et al. [2020] expanded on this finding by removing the input language at test time, thereby reducing the role of language as a auxiliary loss signal, and argued that language serves primarily to regularize learned representations. Distinguishing between descriptive and explanatory language signals, Lampinen et al. [2022] found that the latter uniquely allows models to resolve ambiguity in confounded samples to appropriately assign credit to the causal features. Hase and Bansal [2021] enumerated several ways to incorporate explanation in machine learning models, which is relevant in our context of how to incorporate relation abstraction together with concrete arithmetics.

3 Incorporating Relational Abstraction

In this section, we describe our framework of incorporating relational abstractions into mathematical reasoning. Problem solving can be thought of taking a series of intermediate steps to reach the goal, each of which consists of a numerical expression and an abstract description of the transition between one abstract item to another.

For example, to estimate the total number of students in a school, we begin with the number of grade levels at the school, which is then multiplied by the number of classes in each grade to get the number of classes, and then again multiplied with the number of students in each class to get the total number of students. This relational plan describes the relational reasoning process without invoking any numbers. Here, "number of grades", "number of classes per grade", "number of classes ", "number of students per class" and "number of students" are quantities needed to reach the goal, and the multiplications are the intermediate transition functions. Having formulated the abstracted plan, we can then work out the numerical computations to find the actual quantities.

What makes a plan, such as the one above, abstract is that it omits some pieces of information, such as the quantities involved, and connects items through how they relate to each other. In other words, a relational abstraction formulates the problem as a graph of interconnected items. It may seem that a carefully designed symbolic system can then easily solve the problem by traversing through this graph, but the challenge remains that defining semantic nodes and transitional edges in every problem is difficult. Thus, instead of directly converting plans to graph symbols, we explore the use of structured natural language as a middle ground between symbolic language and unstructured natural language.

To this end, we follow a sequence-to-sequence paradigm as it can be easily adapted to a Trans-

former model. Figure 1 enumerates a few possibilities for how we can incorporate structured relational reasoning into sequence-to-sequence modeling. Assume that we can decompose a solution sequence into the numeric and relational part. Using *numeric-only* formulates the solution by incorporating only numbers and arithmetic operations, which serves as our baseline. In *relational-first*, the relational statements are indicated before numeric ones. This represents the strategy of generating a high-level relational plan first, and then implementing the plan by computing the relevant numbers. Alternatively, the *interleaved* format goes through the relational and numeric steps one after another, alternating between the abstract planning and arithmetic steps. Lastly, in the *multitask* approach, the model is prompted to *either* output the relational *or* the numeric components, but not both, which may allow the model to learn to be implicitly aware of the high level abstraction while writing down the numeric equations. This approach tests the claim that additional auxiliary language tokens effectively function as regularizers or learning tools that can be discarded at test time and may even suppress performance if included [Mu et al., 2020, Lampinen et al., 2022, Hendrycks et al., 2021]. Moreover, learning and generating the two sequences separately has the added advantage of generating shorter sequences at test time, just like numeric-only. In this paper, we examine which type of relational abstraction brings the best reasoning capability.

An alternative to our approach is to allow the model to output only unstructured natural language-based solutions by leveraging existing world knowledge learned elsewhere. This is in fact the approach taken by recent literature of solving mathematics problems [Cobbe et al., 2021, Wang et al., 2022, Wei et al., 2022], and the intent is to invoke the reasoning capability of a pre-trained language model by describing the solution using natural language. Compared to unstructured natural language sequences, our abstract plans encourage the model to extract meaningful concepts and relations to focus on the associated quantities, rather than producing sentences that broadly resemble the training corpus. Separating the relational and numeric segments of the solution can also allow the model to generalize to more problems since many problem instances share the same abstract structure but are filled with different numeric values.

4 Experiments

In this section, we describe the two tasks that we use to test our hypothesis: a set of natural language math problems from the GSM8K dataset [Cobbe et al., 2021] and an abstract unit conversion task. Both tasks share a similar structure in that they contain units and relations that can be represented by a graph, and involve formulating and solving series of numerical equations. While the core of the two tasks are similar, solving word problems requires natural language understanding and general world knowledge (such as the fact that a dozen consists of 12 items, or that the number of eggs increases when it is laid by a chicken but decreases when it is used in baking cookies) whereas the unit conversion task is wholly abstract and is fully solvable using symbol manipulation rules. Together, our two experiments offer both a rich, naturalistic environment with empirical results for broader applicability and a systematic, synthetic environment that reduces mathematical reasoning to its most abstract form, bringing out the advantage of relational abstractions more clearly.

4.1 Task 1: Solving Grade School Math Problems

We first evaluate our framework on more realistic problems posed in natural language as provided by the Grade School Math 8K (GSM-8K) dataset [Cobbe et al., 2021], which contains around 7.7K training question and 1.3K test questions with human annotated solutions, all in the form of the English language. An example of the problem and its solution can be found in the first two rows of Table 1. The original dataset contains the following possible solution formats:

- The *original solution* format provides solution steps and is what was used for the results reported in the original paper. It is based entirely on natural language with portions annotated with executable equations. It shares some similarities with our interleaved approach as it often leaves the target unit of each step at the end of the sentence (e.g. Janet sells 16-3-4 *eggs* a day).

- The *equation-only* format contains the numerical equations without any use of natural language to reference any objects or units.

Problem	Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?
Natural language	
Original solution	(1) Janet sells 16 - 3 - 4 = ≪16-3-4=9≫ 9 duck eggs a day. (2) She makes 9 * 2 = ≪9*2=18≫18 every day at the farmer's market.
Numeric only	
Equation only	(1) ≪16-3-4=9≫ (2) ≪9*2=18≫
Socratic prompts	
Socratic + solution	(1) How many eggs does Janet sell? Janet sells 16 - 3 - 4 = ≪16-3-4=9≫ 9 duck eggs a day. (2) How much does Janet make at the farmers' market? She makes 9 * 2 = ≪9*2=18≫18 every day at the farmer's market.
Socratic + equation	(1) How many eggs does Janet sell? ≪16-3-4=9≫ (2) How much does Janet make at the farmers' market? ≪9*2=18≫
Relational + numeric	
Relation + equation	(1) eggs laid per day - eggs for breakfast - eggs for baking = remaining eggs ≪16-3-4=9≫ (2) remaining eggs * price per egg = amount earned daily from eggs ≪9*2=18≫

Table 1: GSM math dataset sample problem and variants of solution sequence format.

- The *socratic* version contains a series of questions that ask for intermediate answers, which we can either prepend before each step of the original solution (*socratic + solution*), or of the equation-only format (*socratic + equation*). Although the GSM-8K dataset contains these questions as additional content, the original and subsequent works did not use them.

In addition to these formats, we introduce the *relation + equation* format that features relational abstractions. Not only are the resulting states specified, but also the input arguments and the types of transition functions which when taken together, may provide a better form of supervision. For example, "amount earned" is the step output, and "number of eggs multiplied by price per egg" is the relational statement needed to compute the output. Since the original dataset only contains language solutions without any additional labels, we asked human participants to annotate the entire GSM-8K dataset so that each solution step would be paired with an abstract relation. We include our labeling task instructions in the Appendix B. We have also released our collected annotation data to the research community[1].

Both the socratic and relation formats contain pairs consisting of an auxiliary sequence and a solution sequence. Following the setup outlined in Section 3, we either place the auxiliary sequence first or interleave it with the numerical expressions, which we refer to as *aux-first* and *interleaved* respectively in our results. We also include a *multitask* variant of our *relation* format which we describe later.

Implementation. We follow the experimental protocols from Cobbe et al. [2021] and use pre-trained GPT2-M and GPT2-XL models [Radford et al., 2019] for this task. We first fine-tune the model on the question & answer sequences for 40 epochs using a learning rate of 1e-5 with the AdamW optimizer [Loshchilov and Hutter, 2019]. During testing, we also use the calculator pattern to autocomplete the equations as done in Cobbe et al. [2021]. We scan the answer token "####" before the final answer and extract the remaining tokens to compare them against the target answer to report the final accuracy. At test time, we generate output sequences primarily using either

[1] `https://github.com/renmengye/grade-school-math-relational`

Method	GPT2-M (345M)	GPT2-XL (1.5B)	GPT2-XL (1.5B) + Verifier (345M)
Baseline without sequence generation			
Answer only	3.56	4.93	-
Solution sequences only			
Original Solution	10.69	17.44	23.35
Our Equation Only	**15.32**	**22.97**	**24.97**
Auxiliary and solution sequences: Model generates both			
Socratic + Soln. (aux-first)	10.01	13.95	-
Socratic + Soln. (interleaved)	9.93	17.51	-
Socratic + Eqn. (aux-first)	13.27	19.03	23.35
Socratic + Eqn. (interleaved)	15.16	21.00	25.85
Relation + Eqn. (aux-first)	12.59	19.48	25.55
Relation + Eqn. (interleaved)	13.19	22.97	29.49
Auxiliary and solution sequences: Trained to generate either, prompted for numeric at test			
Relation + Eqn. (multitask)	**15.62**	**28.05**	**30.17**
Auxiliary and solution sequences: Prompt with auxiliary, model generates solution sequence			
Socratic + Soln. (aux-first)	17.46	26.23	-
Socratic + Soln. (interleaved)	17.89	28.89	-
Socratic + Eqn. (aux-first)	20.47	35.56	-
Socratic + Eqn. (interleaved)	27.82	36.92	-
Relation + Eqn. (aux-first)	54.59	64.59	-
Relation + Eqn. (interleaved)	**58.53**	**66.26**	-

Table 2: GSM-8K Finetuning Top-1 Test Solve Accuracy (%)

greedy decoding or a verifier model on 20 samples following Cobbe et al. [2021] (see Appendix A.1 for more details). We select the most informative conditions for the verification experiment, due to the lengthy time generating samples for each question. When conditioning on ground-truth auxiliary sequences at test time, we do not perform verification as the same output samples are generated multiple times.

Results. Table 2 shows the main results using GPT2-M and -XL with greedy decoding. The larger language model achieves better performance across the board, though the margin varies with other factors. Note that our numbers are obtained using GPT-2, which is about 100× smaller than GPT-3 in terms of parameter count, so lower accuracy is to be expected.

Compared to the *answer-only* baseline, in which the intermediate steps are omitted, all of the multi-step approaches offer an improvement. Equation-only outperforms the original solution format (22.97% vs. 17.44%), which contains both numbers and text, and this advantage generally holds in other matched comparisons.

When the model is fine-tuned with auxiliary sequences (socratic or relation sequences) paired with solution sequences (either the original GSM8K solution or our numeric equation sequences), we see that performance is no better, and is generally worse, when it must generate both types of sequences than performance with the corresponding solution sequence only. However, interpreting these results is made difficult by the fact that the sequences the model is fine-tuned with are quite long, and performance generally degrades as sequence length increases, creating a potential confound. We find that accuracy generally decreases with increasing solution steps and answer length, and the equation only format suffers the most obvious degradation (see Appendix A.2 for details).

Our multitask regime provides a condition that avoids this difficulty. Here, training sequences contain a prompt for either a relational or a numeric equation sequence, and the model is prompted to generate the equation sequence at test time. We see that the multitask training leads to substantially improved performance in generating the correct equations in the larger GPT2-XL model (28.05% correct compared to the baseline of 22.97%, a 22% relative improvement). This finding shows clearly that training to reason relationally can improve test-time performance, even though at test-time the

model is only generating numerical sequences.

Relation + equation (interleaved) achieves better results than equation-only (29.49% vs. 24.79%), and is almost on par with multitask (29.49% vs. 30.17%) when using 20 samples and the external verifier. We find that verification is less helpful when the output format is purely numeric, such as in the multitask and equation only formats.

We also find that when models trained with auxiliary and solution sequences are prompted at test time with the ground-truth auxiliary sequence for the given problem, model accuracy improves significantly (see Table 2). Strikingly, prompting with ground-truth relational sequences triples the accuracy in the equation-only model (66.26% vs. 22.97%). Moreover, our relational sequences are far better prompts than the GSM8k socratic questions (66.26% vs. 36.92%), suggesting that with a good abstract relational plan, language models can solve the math questions much more easily. These results also indicate that the challenge the models face lies primarily in constructing the correct relational plan.

All else being equal, generating the full relational sequence first as an overall plan is nearly always slightly worse than interleaving relational and equation sequences, and this general pattern holds throughout our results in Tables 2 and 6. The fact that this pattern continues when the relational sequences are provided as prompts suggests that proximity between the corresponding relational and numerical reasoning components helps the model retrieve the correct numeric information.

4.2 Task 2: Unit Conversion

The unit conversion task takes as input a given quantity and unit, then requires finding the equivalent quantity in another unit based on a set of conversion rules that are provided in the prompt (see Table 3). Problems of this type correspond abstractly to a subset of the problem types encountered in GSM8K. The conversion rules are presented in random order, and can collectively be viewed as edges of a graph. Although conversions are bidirectional, only one direction is specified directly in the prompt for each rule so that solving the task is equivalent to finding a path from the source node to the destination node while performing the corresponding multiplication (forward) or division (backward) operations when traversing each edge. This task offers a second context, using totally synthetic problems that eliminate any world knowledge and linguistic uncertainties that the GSM8K problems present, in which to explore the role of teaching the model to identify the abstract sequence of unit conversion steps rather than just step through the required sequence of numeric conversions. In this task setting, we find a very clear advantage from providing and training models to produce relational, as well as numeric, sequences compared to producing numbers alone.

The task is presented as a sequence completion task using the graph description and the conversion instructions as the task prompt (see example in Table 3 under Task Prompt). Depending on the experimental condition, we train the model to produce a target sequence which describes the conversion path traversed and the arithmetic operations involved. Following the general paradigm illustrated in Figure 1, the *relational-plan* approach begins by generating the sequence of units to traverse before producing the sequence of steps containing numeric calculations. As before, the numeric-only approach contains only the arithmetic in each step, whereas the *interleaved* approach includes both the abstract state and the numerical expression in each statement. Additionally, we consider three sub-types for the interleaved approach: *units-then-numbers* states the source and destination units of the traversing edge, followed by the numerical expression; *numbers-then-units* states the numerical expression, followed by the source and destination units; *integrated* states the source quantity and unit, then the remainder of the numerical expression, followed by the destination unit. We refer the reader to Table 3 for examples.

Note that in this task, the relational plan is optional, followed by either numeric-only or one of the three interleaved steps. As in the previous task, we also test each model's capacity to execute a provided correct relational plan by including the ground-truth plan as part of the given prompt for relational plan models. Overall, we have 4 sequence types $\times$ 3 plan types (Ground Truth Plan, Relational Plan, and No Plan) for a total of 12 conditions in comparison. Lastly, as in the previous task, we also consider the multitask approach which we address later in this section.

Implementation. To maintain consistent difficulty across our analyses, we use graphs with 10 nodes and 12 edges, and problems that could be solved using exactly 5 edge traversals. All arithmetic

Task Prompt	**Relational Plan** (Optional)	**Sequence Types**			
		Numeric Only	**Interleaved**		
			Units Then Numbers	Numbers Then Units	Integrated
graph					
H → 2 A F → 3 D	relations	steps	steps	steps	steps
B → 3 A I → 3 F	J → I → F →	1 * 2 → 2	J I 1 * 2 → 2	1 * 2 → 2 J I	1 J * 2 → 2 I
E → 3 B J → 2 I	D → C → G	2 * 3 → 1	I F 2 * 3 → 1	2 * 3 → 1 I F	2 I * 3 → 1 F
B → 3 C F → 4 E		1 * 3 → 3	F D 1 * 3 → 3	1 * 3 → 3 F D	1 F * 3 → 3 D
G → 3 C I → 4 H		3 * 2 → 1	D C 3 * 2 → 1	3 * 2 → 1 C D	3 D * 2 → 1 C
D → 2 C G → 1 B		1 / 3 → 2	C G 1 / 3 → 2	1 / 3 → 2 C G	1 C / 3 → 2 G
convert 1 J to G		<S> 2 G </S>	<S> 2 G </S>	<S> 2 G </S>	<S> 2 G </S>

Table 3: Example of a unit conversion task problem represented in different formats.

operations in this task are performed in modulo-5 to avoid the arbitrary fractions and large numbers that would result from compounding multiplication and division operations involved in multi-step problems. This allows us to focus on the reasoning component of the task rather than the numerical accuracy of performing long arithmetic operations.

We use the same model architecture for all experiments in this task: a 4-layer Transformer Encoder with 4 heads and 512x512 feedforward layers, and a linear token decoder. All intermediary hidden layers are of size 256. We train all models using teacher-forcing on datasets of 10,000 randomly generated problems with 20,000 gradient updates on batches of 256 samples. We measure correctness by extracting the tokens between <S> and </S>, which in fully trained models always consists of 1, 2, 3, or 4 followed by the goal unit, resulting in a 25% chance to correctly guess the answer, even with incorrect intermediary steps.

Method	**Accuracy**
Numeric Only	
Numeric Sequences	25.9 (1.1)
Relational and Numeric	
Relational plan then numeric	69.0 (2.0)
Interleaved: units then numbers	**83.5** (1.5)
Interleaved: numbers then units	69.3 (2.9)
Interleaved: integrated	54.1 (3.0)
Plan + Interleaved: units then numbers	72.5 (2.2)
Plan + Interleaved: numbers then units	74.4 (1.7)
Plan + Interleaved: integrated	77.1 (1.9)
Relational (Prompted) and Numeric	
Relational plan then numeric	84.7 (4.8)
Plan + Interleaved: units then numbers	96.7 (1.2)
Plan + Interleaved: numbers then units	95.5 (2.2)
Plan + Interleaved: integrated	97.6 (1.1)

Table 4: Unit conversion accuracy over 20 runs. Standard errors in parentheses.

Results. All models successfully learned to generate sequences with the corresponding template, but the accuracy of the generated sequences varied from chance to nearly perfect across the conditions we considered. Our findings, summarized in Table 4, demonstrate foremost the importance of having the relational components as part of the target sequence, indicated by the at-chance accuracy of the numeric-only model when trained without planning, and the much higher success rate of all variants including abstract variables (variables corresponding to units).

Of the variants in which the model generates both relational and numeric content at test time, the units-then-numbers model has the highest accuracy. Producing the relational plan first and numeric-only sequences performs slightly weaker, comparable to our findings in Task 1. The fact that units-then-numbers is the best of the three interleaved formats when the model does not first generate a relational plan supports the view that identifying all of the relevant units that need to go in a numeric computation prior to performing that computation can be very helpful.

Although training the model to produce both a relational plan and relational steps interleaved with numbers is helpful in numbers-then-units and integrated conditions, the reverse is true in the units-then-numbers condition, where asking the model to produce an initial relational plan actually reduces accuracy from 83% to 72%. This pattern of results suggests that generating the correct initial relational plan can itself be a challenge, and that an incorrect initial plan then interferes with performing the correct computations.

Consistent with this interpretation, we find that all models trained to produce a relational plan

# Graph Nodes	# Graph Edges	# Solution Steps	Interleaved (UTN)	Numeric Only	Multitask Plan	Multitask Numeric
5	5	2	100.0	94.2	100.0	100.0
6	6	2	100.0	50.0	98.4	89.6
7	8	3	99.0	27.8	85.8	50.2
10	12	5	83.5	25.9	71.6	29.8

Table 5: Unit conversion results by difficulty. **Multitask Plan** indicates the percent of relational plans correctly traversed from the start to goal units by the multitask model. **Multitask Numeric** indicates final answer accuracy in the numeric only outputs by the multitask model. **UTN**: units-then-numbers

do significantly better when given the ground truth plan as part of the prompt, reaching over 95% accuracy in all but the numeric-only models. This suggests that the errors in the planning models are introduced during the abstract planning stage and hurt subsequent performance. In other words, the primary challenge of this task is not performing the correct arithmetic operations, but knowing which steps to take next.

Why is numeric-only so poor? We note that the at-chance performance of the numeric-only format without the relational plan is in contrast to our results in Task 1, as well as some other previous works that solved word problems by mapping them to arithmetic expressions first [Wang et al., 2017, Amini et al., 2019]. We considered whether this was due to the modulo-5 number range used in our experiments, as this would force multiple conversions to use the same numbers. To test this, we trained the numeric-only model on modulo-23 and modulo-53 problems, and also trained the units-then-numbers (interleaved) model on the same problems for comparison. Raising the mudulus increases difficulty, reducing the accuracy of the unit-then-numbers (no plan) model to 71.0% and 31.6% respectively, but numeric-only accuracy drops further to 4.5% and 1.9%, i.e. the expected accuracies for randomly guessing, indicating that the problem persists with larger numeric ranges.

We also consider the possibility that the model may struggle to learn the unit conversion task with numbers-only due to the higher complexity of the task, compared to the GSM8K. Consider the GSM8K problem shown in Table A.1, which requires only a 2-step solution using just 6 unique quantity-unit pairs, and where the quantities invoked in the solution steps appear in the same order as presented in the prompt. In contrast, the graphs used in our analyses contain 10 nodes with 12 edges, and the relations are always presented in random order with no correspondence to how they appear in the solution. These features could make the unit conversion task more difficult, requiring more relational planning.

We test this hypothesis by training the numeric-only and unit-then-numbers (interleaved, no plan) models on three easier datasets that contain problems involving smaller graphs with 5, 6 7 nodes and only 2 to 3 solution steps. We find that while the interleaved unit-then-numbers (UTN in Table 5) models reach near perfect accuracy in all three datasets, the numeric-only models only solve 94.2% of the 5-node, 50% of 6-node, and 28% of 7-node problems (see Table 5). This suggests that while the numeric-only format may work well for simpler problems, it does not scale as well with planning complexity.

Multitask in unit conversion. In Task 1 we find some empirical benefit of the multitask approach, where the model is trained to output either the relational plan or the numerical expressions, then evaluated on the numerical expressions at test time. To answer whether multitasking can be a universal approach, we train 5 models to either produce the relational plan sequence or the numeric-only sequence by prompting it with an 'abstract' or 'steps' at the end of the prompt, similar to Task 1. Here, we find that only 29.8% of the problems are correctly solved at test time, significantly lower than any of the other relational reasoning models (see Table 5). Interestingly, the models produce correct relational plans that connect the start and goal units on 71.6% of the problems, suggesting that the low accuracy in the numeric sequence is not from failing to traverse the graph, but rather from the inability to integrate relational and arithmetic reasoning together.

To examine how multitask performance scales with task complexity, we train the multitask models on easier problems as described in the previous section. On a 5-node graph with 2-step solution problems, the multitask models successfully solve every held-out problem, but this accuracy drops to

50.2% on a 7-node graph with 3-step solution problems. In sum, while the multitask approach with numerical expressions and relational abstractions separately may be effective on simpler problems, this strategy also does not scale well with planning complexity. When the problem is more difficult, it helps to engage in relational abstractions while producing a solution.

5 Discussion

We find that relational reasoning is a key component of mathematical reasoning, whether using natural language or abstract symbols as indicated by our experiments on the GSM8K and the unit conversion tasks. Training the models with relational abstraction can outperform models trained using numerical expressions only, and making these abstractions more salient improves performance further still. While the models can solve some problems without relational abstractions at test time, and can benefit from learning to generate the relational plan separately as in the multitask setup, performing both relational and numerical reasoning together scales far better with model complexity.

We also find that even when all the relational and numerical components are present, how they are ordered makes a significant difference. Among the variants we considered, performing the relational reasoning step just before the numerical computation step is most advantageous, outperforming cases where the full relational plan must be generated at the outset. Lastly, we find that providing the model with the correct abstract steps produces a massive boost in performance, resulting in a 3-fold increase in accuracy for the GSM8K task and near-ceiling accuracy in unit conversion, suggesting that the core of the challenge is indeed correct relational planning.

Some of our other findings will require further work to understand more fully. For example, in GSM8K, equation-only is just as effective as relation + equation when using greedy decoding, and the multitask model far outperforms both, contradicting our findings in the unit conversion task. Although the unit conversion experiments on problems with varying difficulties offer some insight as to why the results differ, it is difficult to be conclusive about why these disagreements occur, especially given the complexities of natural language and the still limited understanding of large language models. For example, language models often struggle with longer sequence lengths, and we cannot claim to know how much of the equation-only model's advantage may be due to shorter solution lengths. This invites further investigation into understanding how large language models learn and represent relational abstractions.

We also recognize the practical constraints of training a model with detailed annotations. Generating the correct labels is difficult to automate and human annotations can be extremely costly. Moreover, determining the right level of abstraction and defining the correct relations can be more art than science, more so when considering more complex domains than math word problems designed for children, which presents an exciting challenge for future research. Indeed, our analyses here are limited to mathematical reasoning (just one form at that), and it is an open question how they will apply to problem solving more broadly.

Ultimately, the road towards building models of general and flexible intelligence requires moving away from curating additional human-labeled data. Our goal is not to find yet another way to engineer supervised learning signals with new types of datasets, but to understand what forms of experiences are most conducive for learning to reason. There is yet much to be understood about reasoning and abstraction in neural networks: *e.g.* 1) what is the scaling relationship between relational abstraction and the ever growing size of large language models and 2) how would a model generate its own abstractions and in domains beyond mathematical reasoning? As a small step towards that end, we hope that this work begins to offer insight into the role of relational abstraction in computational intelligence.

Acknowledgment

We thank Surge AI [2] and the labeling workers for their help creating the abstract auxiliary relation format on GSM-8K.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1245.

Jacob Andreas, Dan Klein, and Sergey Levine. Learning with latent language. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2166–2179. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-1197.

Daniel G Bobrow. Natural language input for a computer problem solving system. 1964.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311, 2022. doi: 10.48550/arXiv.2204.02311.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

Iddo Drori, Sunny Tran, Roman Wang, Newman Cheng, Kevin Liu, Leonard Tang, Elizabeth Ke, Nikhil Singh, Taylor L. Patti, Jayson Lynch, Avi Shporer, Nakul Verma, Eugene Wu, and Gilbert Strang. A neural network solves and generates mathematics problems by program synthesis: Calculus, differential equations, linear algebra, and more. *CoRR*, abs/2112.15594, 2021.

[2] `https://www.surgehq.ai/`

Peter Hase and Mohit Bansal. When can models learn from explanations? A formal framework for understanding the roles of explanation data. *CoRR*, abs/2102.02201, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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. *NeurIPS*, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

Andrew K. Lampinen, Nicholas A. Roy, Ishita Dasgupta, Stephanie Cy Chan, Allison C. Tam, James L. McClelland, Chen Yan, Adam Santoro, Neil C. Rabinowitz, Jane X. Wang, and Felix Hill. Tell me why! explanations support learning relational and causal structure. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 11868–11890. PMLR, 2022.

Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*, 2020.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. *CoRR*, abs/2206.14858, 2022. doi: 10.48550/arXiv.2206.14858.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 158–167. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1015.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 975–984. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.92.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Jesse Mu, Percy Liang, and Noah D. Goodman. Shaping visual representations with language for few-shot classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4823–4830. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.436.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Gabriel Recchia. Teaching autoregressive language models complex tasks by demonstration. *arXiv preprint arXiv:2109.02102*, 2021.

Adam Santoro, Ryan Faulkner, David Raposo, Jack Rae, Mike Chrzanowski, Théophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy Lillicrap. Relational recurrent neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 7310–7321, Red Hook, NY, USA, 2018. Curran Associates Inc.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*, 2019.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1132–1142. The Association for Computational Linguistics, 2015. doi: 10.18653/v1/d15-1135.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed H. Chi, and Quoc Le. Lamda: Language models for dialog applications. *CoRR*, abs/2201.08239, 2022.

Shyam Upadhyay and Ming-Wei Chang. Draw: A challenging and diverse algebra word problem set. number. Technical report, MSR-TR-2015-78, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *CoRR*, abs/2203.11171, 2022. doi: 10.48550/arXiv.2203.11171.

Yan Wang, Xiaojiang Liu, and Shuming Shi. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, 2017.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903, 2022.

Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hanna Hajishirzi, Yejin Choi, and Kyunghyun Cho. Naturalproofs: Mathematical theorem proving in natural language. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.

Method	Greedy	Simple Plurality	Verifier Rerank	Verifier Weighted Plurality
Original Solution	17.44	21.53	19.86	23.35
Our Equation Only	22.97	23.58	23.96	24.79
Socratic + Eqn. (aux-first)	19.03	21.46	22.29	23.35
Socratic + Eqn. (interleaved)	21.00	22.21	25.85	25.47
Relation + Eqn. (aux-first)	19.48	22.97	22.75	25.55
Relation + Eqn. (interleaved)	22.97	26.31	25.63	29.49
Relation + Eqn. (multitask)	**28.05**	**29.42**	**28.28**	**30.17**

Table 6: GSM-8K Top-1 Test Accuracy (%) Using 20 Samples. **Bold** = Best Answer Format; Underline = Best Voting Mechanism. We take results from the best voting mechanism for each method in the main paper.

A Additional results

A.1 GSM-8K results using samples

In Table 6, we study more sample-based mechanisms for generating solutions. We generate 20 samples using softmax sampling (temperature = 0.9), and to aggregate the answers, we considered plurality voting [Wang et al., 2022] and the following verification-based techniques:

- **Verification.** As originally proposed in Cobbe et al. [2021], we train a separate verifier model using samples generated by our main model. The verifier takes as input the concatenated sequence of question and answer, then outputs a sequence of scores predicting whether the answer is correct or not. We generate the training samples using the main model after two epochs of fine-tuning, then fine-tune the GPT2-M model as our verifier.
- **Verifier weighted plurality.** We find that as the number of samples grows, a simple reranking mechanism performs worse as it has more incorrect options to choose from as the top choice. Cobbe et al. [2021] proposes using the voting mechanism to select the top-K ranked samples as seeds and voting among these candidates. However, this requires a larger number of samples for the voting process, and moreover, K becomes yet another hyperparameter to tune. Here, we explore a simpler approach of using the verifier score to weigh the votes. We find that it smooths out predictions and achieves higher accuracy.

All models seem to improve with using 20 samples, and our verifier weighted plurality is the best approach, achieve the best overall accuracy on all but one condition. Figure 2 and 3 show accuracy as a function of number of samples, and the verifier weighted plurality achieves higher scores with more samples.

Table 6 also indicates that performance of verification-based approaches benefits more from additional auxiliary information (whether in the form of natural language or abstract relations). For instance, our proposed *relation + equation (interleaved)* format has a similar performance to *equation only* using greedy decoding, but achieves significantly better performance with a verification voting procedure, while *equation only* receives a smaller boost (interleaved improved by +6.52% vs. equation only +1.82%). The *original solution* also receives a boost of +5.91%, except that the absolute accuracy is 6.14% lower than *relation + equation (interleaved)*, a rather wide gap. This dependence on a verification plus voting procedure suggests that relational abstraction is a more computationally demanding task that requires repeated processing of information.

A.2 GSM-8K results on different solution length

In Figure 2 and Figure 3 we show the accuracy as a function of number of samples in both reranking and weighted plurality voting schemes. Reranking sometimes suffers from lower accuracy with more number of samples, whereas weighted voting has an overall positive trend as the number of samples go up.

We compare the performance of problems with different numbers of solution steps (Figure 4) and different generated sequence lengths (Figure 5). The overall trend confirms that models per-

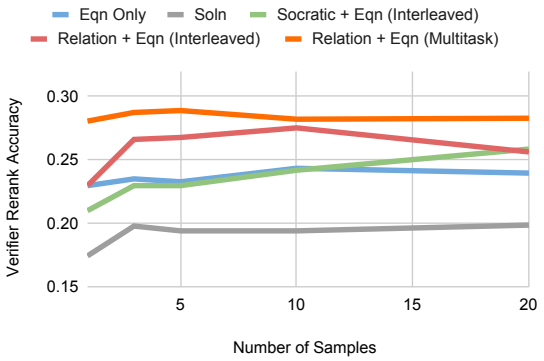

Figure 2: Verifier reranking accuracy

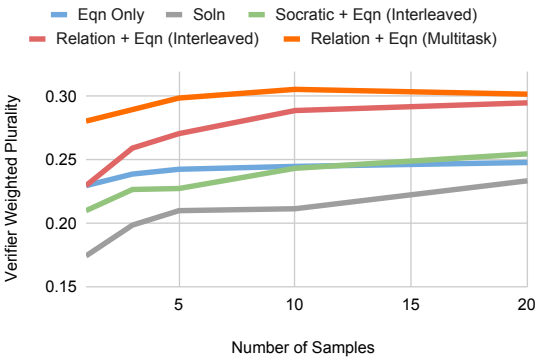

Figure 3: Verifier weighted plurality accuracy

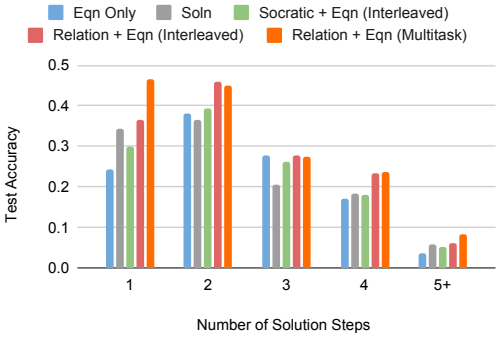

Figure 4: Accuracy vs. number of reasoning steps in the groundtruth answer.

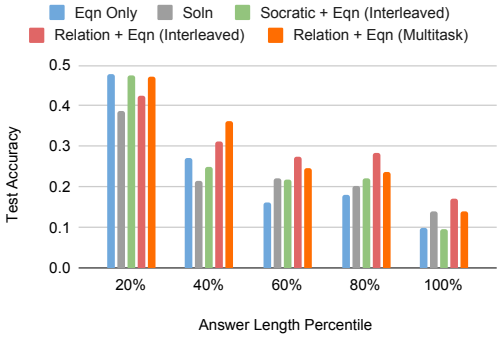

Figure 5: Accuracy vs. percentile of solution length (percentiled separately by condition).

form worse with longer answers. Figure 5 suggests that Equation Only tends to suffer from more degradation as the relative solution length increases.

B Human annotator instructions

We include our instruction for human annotators for collecting the abstract relational plan data for GSM-8K dataset. The following pages contain an instruction as well as an example to be annotated with empty fillable boxes. This shows the user interface that the human annotators used when the labeling task was performed.

Instruction

You will be assigned with some grade school math questions. The full solution is provided below each question. For most steps in the solution, there is a math equation being highlighted. Please add a line of *explanatory text* for each equation. The explanation should follow the *same format* as the original equation, while describing the items with short phrases that connect the equation with the relevant quantities mentioned in the problem and with quantities computed in other problems. Try to construct phrases that characterize the quantities succinctly while avoiding ambiguity and use the same phrase to refer to the same quantity a second time.

Here are some example questions. The purple text below illustrates the kinds of phrases that we ask you to fill in:

Example #1

Question: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

Solution: Janet sells 16 - 3 - 4 = <<16-3-4=9>>9 duck eggs a day.
She makes 9 * 2 = $<<9*2=18>>18 every day at the farmer's market.

Final answer: 18.

Line 1: 16-3-4=9 **Explanation:** Eggs laid - eggs eaten - eggs baked = eggs sold
Line 2: 9*2=18 **Explanation:** Eggs sold * price per egg = amount earned

Note that we have preferred the use of very general names for variables such as "price" rather than "dollars" to encourage the recognition of common structures of variables. We also used the exact phase 'eggs sold' both for the result of line one and for the same quantity when it occurred on the left hand side in line 2.

Please try to explain all quantities in the equations, including the item after the "=" sign.

Please also have white space before and after mathematical symbols like "+", "-", "", "/", "=", etc.*

Example #2

Question: Jen is planning to sell her root crops. She has 6 yams which can be sold at $1.5 each, 10 sweet potatoes that cost $2 each, and 4 carrots which cost $1.25 each. If she sells everything, how much will she earn?

Solution: Jen can earn $1.5 x 6 = $<<1.5*6=9>>9 for the yams.
She can earn $2 x 10 = $<<2*10=20>>20 for the sweet potatoes.
And she can earn $1.25 x 4 = $<<1.25*4=5>>5 for the carrots.
Therefore, she will earn $9 + $20 + $5 = $<<9+20+5=34>>34 if she sells everything.
Final answer: 34.

Line 1: 1.5*6=9 **Explanation:** Price per yam * number of yams sold = amount earned on yams
Line 2: 2*10=20 **Explanation:** Price per sweet potato * number of sweet potatoes sold = amount earned on sweet potatoes
Line 3: 1.25*4=5 **Explanation:** Price per carrot * number of carrots sold = amount earned on carrots
Line 4: 9+20+5=34 **Explanation:** Amount earned on yams + amount earned on sweet potatoes + amount earned on carrots = total amount earned

Note that we don't repetitively mention the person's name (Jen) since it does not help resolve any ambiguity by mentioning her name.

Example #3

Question: John had a son James when he was 19. James is now twice as old as his sister Dora, who will turn 12 in 3 years. How old will John's youngest son, who was born when John was 32, be in 3 years?

Solution: Dora is 12-3=<<12-3=9>>9.
So James is 9*2=<<9*2=18>>18 years old
That means John is 18+19=<<18+19=37>>37
Johns youngest son is 37-32=<<37-32=5>>5 years old
So he will be 5+3=<<5+3=8>>8 in 3 years
Final answer: 8.

Line 1: 12-3=9 **Explanation 1:** Dora's age in three years - three years = Dora's age now
Line 2: 9*2=18 **Explanation 2:** Dora's age * ratio of James' age to Dora's age = James' age
Line 3: 18+19=37 **Explanation 3:** James' age + John's age when James was born = John's age
Line 4: 37-32=5 **Explanation 4:** John's age - John's age when his youngest son was born = John's youngest son's age
Line 5: 5+3=8 **Explanation 5:** John's youngest son's age + three years = John's youngest son's age in three years

In this example, it is necessary to mention people's names to avoid ambiguity. Only do this when necessary.

Example #4

Question: Every hour Joanne has to collect the coins out of the fountain inside the mall. During the first hour, she collected 15 coins. For the next two hours, she collected 35 coins from the fountain. In the fourth hour, she collected 50 coins from the fountain but she gave 15 of them to her coworker so she could buy a soda. How many coins did she have after the fourth hour?

Solution: 15 coins collected in hour one
35 coins collected in hour two
35 coins collected in hour three
50 coins collected in hour four
Before giving her coworker some coins there were 15+35+35+50=<<15+35+35+50=135>>135 coins
The number of coins after given 15 to her coworker is 135-15=<<135-15=120>>120
Final answer: 120.

Line 1: 15 coins collected in hour one **Explanation:** Coins collected in hour one
Line 2: 35 coins collected in hour two **Explanation:** Coins collected in hour two
Line 3: 35 coins collected in hour three **Explanation:** Coins collected in hour three
Line 4: 50 coins collected in hour four **Explanation:** Coins collected in hour four

Line 5: 15+35+35+50=135 **Explanation:** Coins collected in hour one + coins collected in hour two + coins collected in hour three + coins collected in hour four = total coins collected.
Line 6: 135-15=120 **Explanation:** Total coins collected - coins given to the coworker = coins remaining

Note that not all lines will contain an equation, and in this case try to explain each solution line with plain words.

In some cases, as with the first four lines here, the explanation may repeat the content of the Line, but we ask you to provide such explanations, as in the example.

Equations with unknown variables

For each problem, before you can enter explanations, there will be a **required question** asking whether any of the lines of the solution contain unknown variables. In the example below, "C" is the unknown variable. If there are unknown variables, then please answer "yes" for the first question, and follow the example below to provide an explanation for each line.

Question: Farmer Brown has 20 animals on his farm, all either chickens or cows. They have a total of 70 legs, all together. How many of the animals are chickens?

Solution: Let C be the number of chickens.
There are 20-C cows.
The cows have 4*(20-C) legs.
The chickens have 2C legs.
The total number of legs is 2C+4(20-C)=70.
2C+80-4C=70
2C=10
C=<<5=5>>5.
Final answer: 5.

In this case, you will be asked to provide explanations for each line of the solution which will be displayed. These lines will not simply be an equation as in other cases. As before, the purple text shows the kind of explanation we are asking you to provide.

Line 1: Let C be the number of chickens. **Explanation 1:** Define a variable for the number of chickens
Line 2: There are 20-C cows. **Explanation 2:** Number of animals - number of chickens =

number of cows

Line 3: The cows have 4*(20-C) legs. **Explanation 3:** Number of legs each cow * number of cows = number of cow legs

Line 4: The chickens have 2C legs. **Explanation 4:** Number of chicken * legs per chicken = number of chicken legs

Line 5: The total number of legs is 2C+4(20-C)=70. **Explanation 5:** Number of chicken legs + number of cow legs = total number of legs

Line 6: 2C+80-4C=70. **Explanation 6:** Simplify toward finding the number of chickens

Line 7: 2C=10. **Explanation 7:** Combine like terms toward finding the number of chickens

Line 8: 5=5 **Explanation 8:** Divide by 2 to determine the number of chickens

Note that we have asked you to restate the quantity referenced by the variable and also to use the quantity, not the variable itself in your explanations.

Collapse Instructions

Question: Cynthia has four times as many water balloons as her husband, Randy. Randy has only half as many water balloons as his daughter, Janice. If Janice throws all 6 of her water balloons at her father, how many water balloons does Cynthia have, which she could also choose to throw at Randy?

Solution: Randy has only half as many water balloons as Janice's 6, for a total of (½)*6=3 water balloons.
Cynthia has 4 times as many water balloons as Randy, for a total of 4*3=<<4*3=12>>12 water balloons
Final answer: 12

Does the solution seem correct?
○Yes
○No

Does the solution contain equations of unknown variables? (See instruction for an example of equations of unknown variables)
○Yes
○No

(If the line is empty, please skip the response)

Line 1: Randy has only half as many water balloons as Janice's 6, for a total of (½)*6=3 water balloons.

Explanation 1:

(If the line is empty, please skip the response)

Line 2: 4*3=12

Explanation 2:

(If the line is empty, please skip the response)

Line 3:

Explanation 3:

(If the line is empty, please skip the response)

Line 4:

Explanation 4:

(If the line is empty, please skip the response)

Line 5:

Explanation 5:

(If the line is empty, please skip the response)

Line 6:

Explanation 6:

(If the line is empty, please skip the response)

Line 7:

Explanation 7:

(If the line is empty, please skip the response)

Line 8:

Explanation 8:

(If the line is empty, please skip the response)

Line 9:

Explanation 9:

(If the line is empty, please skip the response)

Line 10:

Explanation 10:

(If the line is empty, please skip the response)

Line 11:

Explanation 11:

(If the line is empty, please skip the response)

Line 12:

Explanation 12:

Additional comments