Learning to Reason With Relational Abstractions

Andrew J. Nam*
Stanford University
ajhnam@stanford.edu

Mengye Ren*
NYU
mengye@cs.nyu.edu

Chelsea Finn
Stanford University
cbfinn@stanford.edu

James L. McClelland
Stanford University
jlmcc@stanford.edu

Abstract

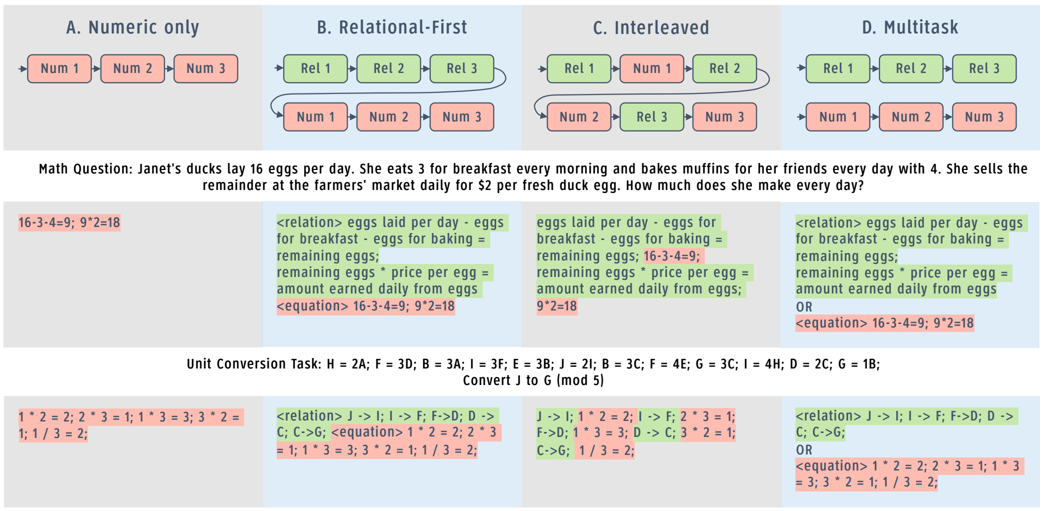
Large language models have recently shown promising progress in mathematical reasoning when fine-tuned with human-generated sequences walking through a sequence of solution steps. However, the solution sequences are not formally structured and the resulting model-generated sequences may not reflect the 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

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 [8, 7, 3, 2]. 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 [2, 3] which introduces further variance depending on the vocabulary and sentence structure, some avoid natural language altogether [9, 5], and some begin with natural language inputs but translate them into purely arithmetic operations while solving [10, 1]. 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 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

2nd MATH-AI Workshop at NeurIPS 2022.



Math Question: Janet's ducks lay 16 eggs per day. She eats 3 for breakfast every morning and bakes muffins for her friends every day with 4. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much does she make every day?

A. Numeric only	B. Relational-First	C. Interleaved	D. Multitask
16-3-4=9; 9*2=18	<relation> eggs laid per day - eggs for breakfast - eggs for baking = remaining eggs; remaining eggs * price per egg = amount earned daily from eggs <equation> 16-3-4=9; 9*2=18	eggs laid per day - eggs for breakfast - eggs for baking = remaining eggs; 16-3-4=9; remaining eggs * price per egg = amount earned daily from eggs; 9*2=18	<relation> eggs laid per day - eggs for breakfast - eggs for baking = remaining eggs; remaining eggs * price per egg = amount earned daily from eggs OR <equation> 16-3-4=9; 9*2=18

Unit Conversion Task: H = 2A; F = 3D; B = 3A; I = 3F; E = 3B; J = 2I; B = 3C; F = 4E; G = 3C; I = 4H; D = 2C; G = 1B;
Convert J to G (mod 5)

A. Numeric only	B. Relational-First	C. Interleaved	D. Multitask

1 * 2 = 2; 2 * 3 = 1; 1 * 3 = 3; 3 * 2 = 1; 1 / 3 = 2;	<relation> J -> I; I -> F; F->D; D -> C; C->G; <equation> 1 * 2 = 2; 2 * 3 = 1; 1 * 3 = 3; 3 * 2 = 1; 1 / 3 = 2;	J -> I; 1 * 2 = 2; I -> F; 2 * 3 = 1; F->D; 1 * 3 = 3; D -> C; 3 * 2 = 1; C->G; 1 / 3 = 2;	<relation> J -> I; I -> F; F->D; D -> C; C->G;
			OR
			<equation> 1 * 2 = 2; 2 * 3 = 1; 1 * 3 = 3; 3 * 2 = 1; 1 / 3 = 2;

Figure 1: How can we incorporate structured relational reasoning in sequence-to-sequence modeling? We outline several ways to combine relational and numeric part of the reasoning process.

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.

2 Incorporating Relational Abstraction

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. 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 Transformer 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 [6, 4, 3]. 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.

3 Experiments

3.1 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 [2], 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.
- 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?
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.
Equation only	(1) ≪16-3-4=9≫ (2) ≪9*2=18≫
Auxiliary (optional)	
Socratic	(1) How many eggs does Janet sell? (2) How much does Janet make at the farmers' market?
Relation	(1) eggs laid per day - eggs for breakfast - eggs for baking = remaining eggs (2) remaining eggs * price per egg = amount earned daily from eggs

Table 1: GSM math dataset sample problem and example solution components

Method	**GPT2-M**	**GPT2-XL**
Baseline without sequence generation		
Answer only	3.56	4.93
Solution sequences only		
Original Solution	10.69	17.44
Our Equation Only	**15.32**	**22.97**
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
Socratic + Eqn. (interleaved)	15.16	21.00
Relation + Eqn. (aux-first)	12.59	19.48
Relation + Eqn. (interleaved)	13.19	22.97
Relation + Eqn. (multitask)	**15.62**	**28.05**
Auxiliary and solution sequences: Prompt with true auxiliary		
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 (%)

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

- 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*).

In addition to these formats, we introduce the *relation* + *equation* format that features relational abstractions. Not only are the resulting states indicates, but also the input arguments and the types of transition functions which we hypothesize provide a better form of supervision. We asked human participants to annotate the entire GSM-8K dataset so that each solution step would be paired with an abstract relation.

Results. Table 2 shows the main results. 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.

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 it is only generating numerical sequences.

Relation + equation (interleaved) is 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

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

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.

3.2 Task 2: Unit Conversion

The unit conversion task involves converting a given quantity and unit, then finding the equivalent quantity in another unit based on the conversion rules that are provided in the prompt (see Table 4). Problems of this type correspond abstractly to a subset of the problem types encountered in GSM8K.

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 4 for examples.

Results. 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. Numeric only is worse here, since 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. These features could make the unit conversion task more difficult, requiring more relational planning.

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. Here, in contrast to the GSM8K results, we find that only 29.8% of the problems are correctly solved by multitask, significantly lower than any of the other relational reasoning models. We hypothesize that while multitask learning with numerical expressions and relational abstractions separately may be effective on simpler problems, this strategy also does not scale well with planning complexity.

4 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. 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 GSM task and near-ceiling accuracy in unit conversion, suggesting that the core of the challenge is indeed correct relational planning.

Acknowledgments and Disclosure of Funding

This work was supported by Stanford Institute for Human-Centered Artificial Intelligence (HAI). We thank Surge AI [1] and the labeling workers for their help creating the abstract auxiliary relation format on GSM-8K.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

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

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

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

[10] 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.