ATHENA: Mathematical Reasoning with Thought Expansion

JB. Kim[1] **Hazel H. Kim**[2] **Joonghyuk Hahn**[1] **Yo-Sub Han**[1]
[1]Yonsei University [2]Classting AI Research
jb@thejb.net, hazel.kimh@gmail.com, {greghahn,emmous}@yonsei.ac.kr

Abstract

Solving math word problems depends on how to articulate the problems, the lens through which models view human linguistic expressions. Real-world settings count on such a method even more due to the diverse practices of the same mathematical operations. Earlier works constrain available thinking processes by limited prediction strategies without considering their significance in acquiring mathematical knowledge. We introduce **A**ttention-based **TH**ought **E**xpansion **N**etwork **A**rchitecture (**ATHENA**) to tackle the challenges of real-world practices by mimicking human thought expansion mechanisms in the form of neural network propagation. A thought expansion recurrently generates the candidates carrying the thoughts of possible math expressions driven from the previous step and yields reasonable thoughts by selecting the valid pathways to the goal. Our experiments show that ATHENA achieves a new state-of-the-art stage toward the ideal model that is compelling in variant questions even when the informativeness in training examples is restricted.[1]

1 Introduction

Math word problem (MWP) solving is one of the fundamental reasoning tasks of answering a mathematical question by understanding a complex, intricate system of human lexical expressions. Models' ability to solve a problem depends on a method that articulates the problem, the lens through which they view human lexical expressions. Ideal MWP models understand the diverse applications of the same mathematical operations in real-world situations, which require lexically sophisticated. For example, "×" can count all elements equally divided in multiple boxes, but also calculate area from length and width, or tax fee from the tax rate and income.

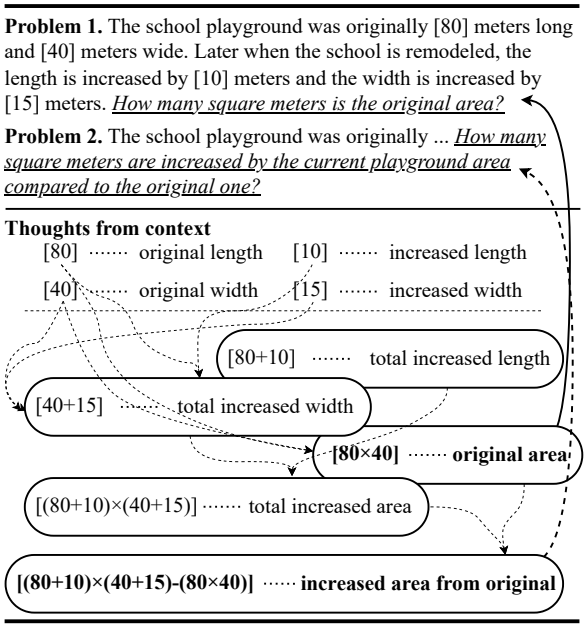

Figure 1: Visualization of thoughts constructed for solving two problem samples with the same context description from the UnbiasedMWP dataset, one of our benchmarks.

It is significant how we estimate if the model has learned mathematical reasoning to qualify for the ideal model. We state that the ideal models that learn mathematical problems must be able to solve unseen problems if they are applications of mathematical operations that models have already seen, or soundly solve problems even when given examples to learn are restricted.

Humans learn mathematical knowledge by formulating and understanding underlying principles from seen cases rather than just recognizing the common lexical patterns. Models currently face two challenges to reach human-level mathematical understanding: *conceptual knowledge* to understand the practices of mathematical principles, and *procedural knowledge* to deductively derive the answer through the principles, which are indispensable for each other in their development and

[1]The source code is available at https://github.com/the-jb/athena-math.

Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 16315–16327
December 6-10, 2023

Context The school playground was originally [80] meters long and [40] meters wide. Later when the school is remodeled, the length is increased by [10] meters and the width is increased by [15] meters.

Train on an example of a question-solution pair under the context above.

Question How many square meters is the original playground area? **Solution** (80×40)

Test on variant questions that share the context above.

Q0 How many times the length of the original playground was the width?

DeductReasoner	**ATHENA**
UnbiasedMWP $(80 + 10) \times (40 + 15) - (80 \times 40)$ (X)	UnbiasedMWP $(80 + 10) \times (40 - 15)$ (X)
UnbiasedMWP (1:N) $80 \div 40$ (O)	UnbiasedMWP (1:N) $80 \div 40$ (O)

Q1 How many square meters is the current playground area?

DeductReasoner	**ATHENA**
UnbiasedMWP $(80 + 10) \times (40 + 15) - (80 \times 40)$ (X)	UnbiasedMWP $(80 + 10) \times (40 + 15)$ (O)
UnbiasedMWP (1:N) 80×40 (X)	UnbiasedMWP (1:N) $(80 + 10) \times (40 + 15)$ (O)

Q2 How many square meters are increased by the current playground area compared to the original one?

DeductReasoner	**ATHENA**
UnbiasedMWP $(80 + 10) \times (40 + 15) - (80 \times 40)$ (O)	UnbiasedMWP $(80 + 10) \times (40 + 15) - (80 \times 40)$ (O)
UnbiasedMWP (1:N) 80×40 (X)	UnbiasedMWP (1:N) $(80 + 10) \times (40 + 15) - (80 \times 40)$ (O)

An example with a lexically similar context to that of above from the UnbiasedMWP

Context The school basketball court was [20] meters long and [12] meters wide. After the renovation, the length is increased by [8] meters, and the width increases by [3] meters.
Question How many square meters are increased? **Solution** $(20 + 8) \times (12 + 3) - (20 \times 12)$

Table 1: Predictions of DeductReasoner (Jie et al., 2022) and ATHENA on a sample that has variant questions while sharing the common context for the problems. The observation above is when models use RoBERTa-large on UnbiasedMWP (Yang et al., 2022).

usage (Rittle-Johnson and Alibali, 1999; Byrnes and Wasik, 1991; Canobi, 2009; Rittle-Johnson and Schneider, 2014).

Prior approaches mostly adopt the transduction-based models such as sequence-to-sequence (Ling et al., 2017; Wang et al., 2018), sequence-to-tree (Xie and Sun, 2019; Liu et al., 2019a) or graph-to-tree methods (Zhang et al., 2020b; Li et al., 2020) and concentrate on enhancing problem-level encoding (Shen and Jin, 2020; Zhang et al., 2020b; Lin et al., 2021; Yu et al., 2021). These works have limitations in obtaining procedural knowledge due to their prediction strategies that operate in a counter-intuitive order. For instance, sequence-to-tree approaches determine mathematical operations before the operands in the inference steps. Recently, Jie et al. (2022) proposed a deductive prediction strategy, but it still entails procedural bias, accepting only one particular reasoning pathway.

Overall, the prior studies show limitations in learning mathematical procedures, which is significant for achieving successful mathematical skills. As a result, despite their high accuracies on some benchmarks, the current approaches fail to solve variant questions that are simply mutated from already trained examples (Patel et al., 2021; Yang et al., 2022). As shown in Table 1, we empirically argue that they learn the repeated patterns in given problems rather than the underlying principles formulating the equations.

The cognitive inadequacy of previous models motivates us to propose a new reasoning architecture that maximizes feasible reasoning pathways. We design ATHENA that reasons with thought expansion inspired by the studies of human reasoning (Johnson-Laird, 2008; Rittle-Johnson and Schneider, 2014). The key idea is to implement two types of thoughts in the expansion process: *thoughts before considering goals* formed by conceptual knowledge and *goal-directed thoughts* yielded by procedural knowledge.

Figure 1 illustrates an example of thoughts formed via asking different questions (goal) within the same situation (context). Because it is tricky for models to answer different questions that share a lexically similar problem context, we state that the two thinking strategies would lead to the right answer by properly utilizing mathematical knowledge. We expand the thoughts by applying conceptual knowledge to obtain candidate thinking pathways and procedural knowledge to evaluate the potential answers. This is how we endow models with mathematical reasoning ability and empirically demonstrate that the model has actually learned the mathematical knowledge.

ATHENA puts the aforementioned thinking pro-

cess explicitly into neural network propagation. Defining the term *thought* as a representation of each math expression driven from the problem, we shape *candidate thoughts* and the goal-directed thoughts named *reasonable thoughts*. The model generates candidate thoughts by applying mathematical operations and yields reasonable thoughts by filtering with solidly updated premises until it meets the appropriate answer. With this recurrent process, we develop a neural model of processing thoughts based on multi-head attention (Vaswani et al., 2017) that effectively carries the subtle feature changes during expansion.

Our experiments show that the proposed approach is strong at predicting mathematical expressions requiring sophisticated comprehension as shown in Table 1. We observe that ATHENA produces a solid performance when the model needs to deal with previously unseen questions. ATHENA is also very compelling to solve variant questions once it has learned one question established from the shared context. From the experimental results, we conclude that ATHENA reaches a new state-of-the-art stage toward the ideal MWP model that we define as the one that can learn mathematical reasoning.

2 Math Word Problem

Math word problem (MWP) solving is the task of answering a mathematical question by understanding natural language descriptions.

2.1 Problem Formulation

Our task of solving MWPs is defined as follows. Each example in the MWP dataset $\mathcal{D}$ has a problem sequence S in natural language as input and an equation $\mathcal{E}$ as expected output. $\mathcal{D}$ consists of K (problem, equation) tuples, where K is the number of examples:

$$\mathcal{D} = \{(S_{(i)}, \mathcal{E}_{(i)})\}_{i=1,\ldots,K}.$$

We use a pre-trained language model (PLM) to embed S. Let $P = (t_1, t_2, \ldots, t_n)$ denote a tokenized sequence of S, where t_i represents each subword token. The PLM output of P is denoted as $X = (x_1, x_2, \ldots, x_n)$, where x_i is an embedding vector of each token t_i.

2.2 Related Work

MWP problems have begun with feature engineering via hand-crafted rules or statistical concepts (Bakman, 2007; Hosseini et al., 2014; Mitra and Baral, 2016). Early works have adopted neural network approaches through end-to-end learning strategies such as sequence-to-sequence (Ling et al., 2017; Wang et al., 2018; Li et al., 2019) or sequence-to-tree (Xie and Sun, 2019; Liu et al., 2019a; Chiang and Chen, 2019; Qin et al., 2020). The previous approaches often use the networks or manipulate the representation with tree or graph templates to generate mathematical equations in a structurally sophisticated manner (Wang et al., 2017; Zhang et al., 2020b).

Having developed and become accessible to pre-training and transfer learning, several approaches have promoted their performance with pre-trained language models (Shen et al., 2021; Yu et al., 2021; Huang et al., 2021; Zhang et al., 2020a; Liang et al., 2022), aiming to enhance the encoder with pre-trained embeddings. Other approaches have made a key contribution by utilizing additional knowledge such as semantic meaning. Some take advantage of structural information such as hierarchical dependency (Shen and Jin, 2020; Lin et al., 2021; Yu et al., 2021), formula structure (Huang et al., 2020), graph-edge connection information (Zhang et al., 2020b; Wu et al., 2021; Li et al., 2020) and more (Li et al., 2022; Shen et al., 2021).

All these approaches mostly aim at enhancing problem-level information but the recent studies demonstrate the significance of inference procedures in reasoning tasks. Wei et al. (2022) show impressive success for large-scale language models in complex reasoning tasks by adopting chain-of-thought prompting. The reasoning extraction method (Jie et al., 2022) has recently reached decent performance by constructing the deductive order in solving MWP.

3 ATHENA

Attention-based **TH**ought **E**xpansion **N**etwork **A**rchitecture (ATHENA) is an architecture that expands its thoughts to solve MWP. Figure 2 illustrates an overall process of ATHENA. ATHENA extracts initial thoughts Θ_0 from PLM and expands them with inferring through premises until it reaches the final thought. We first clarify what is a *thought*—a foundational ingredient of our model—and explain the premise and goal vectors that measure the thoughts.

Thought. A thought is an embedding of a possible math expression derived from quantities in a problem representing the contextual meaning of the

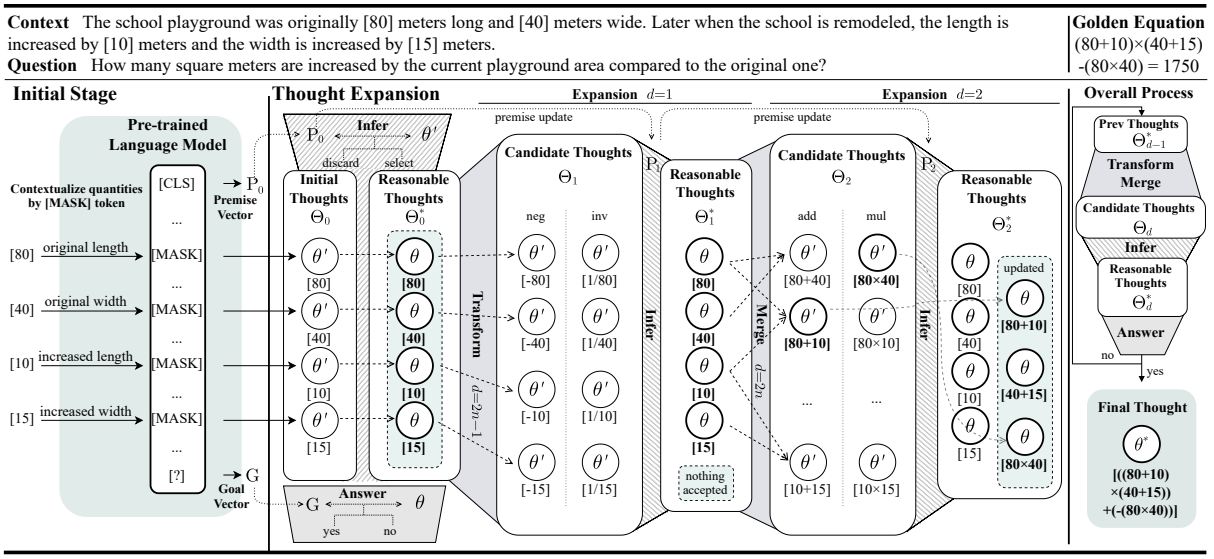

Figure 2: Overall process of ATHENA. First, extract initial thoughts, an initial reasoning vector, and a goal vector from PLM. Second, expand thoughts by transform ($d = 1, 3, 5, \dots$) or merge ($d = 2, 4, 6, \dots$) and generate candidate thoughts. Third, infer the candidate thoughts to obtain new reasonable thoughts. Last, give reasonable thoughts to the next expansion. Repeat until meeting a thought that answers the goal vector.

expression. Let θ denote a thought with hidden size H corresponding to an expression $\mathcal{E}(\theta)$. A goal of the model is to find a thought θ^* that satisfies the ground-truth expression $\mathcal{E}^*$:

$$\mathcal{E}(\theta^*) \equiv \mathcal{E}^*.$$

Premise Vector. A premise vector represents previously inferred thoughts to evaluate and filter candidate thoughts in each depth. Let P_d denote a premise vector for depth d. We set an initial premise vector P_0 with the [CLS] token from the problem descriptions.

Goal Vector. A goal vector plays a role as ground-truth measurement to evaluate if a thought is an appropriate answer to the question. We set a goal vector G with a tokenized embedding of the punctuation mark in the question description.

3.1 Initial Thought

An initial thought is an embedding that carries each quantity representation illustrated in a context or question description. We mask quantities with [MASK] token and obtain the embeddings that capture contextual information from the perspective of corresponding quantities. We denote a set of thoughts in the initial depth by Θ_0:

$$\Theta_0 = \{x_i \mid x_i \in X, t_i \in P, t_i = [\mathrm{MASK}]\}.$$

Certain quantity representations such as π are necessary for generating mathematical expressions despite not being presented in the contexts or questions. We collect them from a training set and randomly initialize their embeddings. We also put their embeddings to initial thoughts Θ_0.

3.2 Thought Expansion

In each depth, thought expansion constructs candidate thoughts Θ_d and filters them to obtain the *reasonable* thoughts Θ_d^*. Reasonable thoughts are the waypoint thoughts to reach the final thought.

The two stages in a thought expansion are: (1) our model generates candidate thoughts Θ_d from previous thoughts Θ_{d-1}^* through the operations and (2) it reasons about the candidates if they are worth to be reasonable thoughts Θ_d^*. Expansion keeps going until finding one of the reasonable thoughts qualified to be a final thought θ^*.

3.2.1 Candidate Thought

Our model generates a set of possible new thoughts Θ_d from the previous thoughts Θ_{d-1}^* as the candidates. A new thought θ' is a thought of a math expression obtained by combining two previous thoughts $\theta_i, \theta_j \in \Theta_{d-1}^*$ with an arithmetic operation:

$$\mathcal{E}(\theta') = \mathcal{E}(\theta_i) \circ \mathcal{E}(\theta_j) \text{ where } \circ \in \{+, -, \times, \div\}.$$

To make a new thought, we introduce two operation layers whose combination can represent the arithmetic operations: merge M and transform T. These layers aim to maximize the reflection of the

characteristics of arithmetic operations rather than the separate layers of individual arithmetic operations. The definitions of merge and transform are shown below.

Merge. Merge layer M merges a pair of thoughts (θ_i, θ_j) into a new thought θ' such that $\mathcal{E}(\theta')$ applies addition and multiplication to $\mathcal{E}(\theta_i)$ and $\mathcal{E}(\theta_j)$:

$$\overset{\mathrm{op}}{\mathrm{M}} : \theta_i, \theta_j \mapsto \theta'$$
$$\text{s.t. } \mathcal{E}(\theta') = \mathrm{op}(\mathcal{E}(\theta_i), \mathcal{E}(\theta_j)) \text{ where } \mathrm{op} \in \{+, \times\}.$$

Transform. Transform layer T transforms a thought θ into a new thought θ' such that $\mathcal{E}(\theta')$ applies inverse operations of addition and multiplication to $\mathcal{E}(\theta)$:

$$\overset{\mathrm{op}}{\mathrm{T}} : \theta \mapsto \theta'$$
$$\text{s.t. } \mathcal{E}(\theta') = \mathrm{op}(\mathcal{E}(\theta)) \text{ where } \mathrm{op} \in \{-\cdot, \cdot^{-1}\}.$$

We use Feed-Forward Network (FFN) and multi-head attention inspired by Vaswani et al. (2017) for the implementation of the operation layers. We use FFN referred to as FF for transform layer T. Using multi-head self-attention $\mathrm{A}_{\mathrm{self}}$ and layer normalization ℓ, we implement merge layer $\mathrm{M}(\theta_i, \theta_j)$ followed by:

$$\mathrm{M}(\theta_i, \theta_j) = \mathrm{FF}(\theta_i + \theta_j + \ell(\mathbf{1}_2^\top \underset{\mathrm{self}}{\mathrm{A}}([\theta_i; \theta_j]))W + b)$$
$$\text{where } W \in \mathbb{R}^{H \times H}, b \in \mathbb{R}^H.$$

This implementation satisfies M^{op} to be commutative for $\mathrm{op} \in \{+, \times\}$:

$$\overset{\mathrm{op}}{\mathrm{M}}(\theta_i, \theta_j) = \overset{\mathrm{op}}{\mathrm{M}}(\theta_j, \theta_i) \text{ and}$$
$$\mathcal{E}(\overset{\mathrm{op}}{\mathrm{M}}(\theta_i, \theta_j)) = \mathcal{E}(\overset{\mathrm{op}}{\mathrm{M}}(\theta_j, \theta_i)).$$

We apply transform layer T for depth $d = 2n - 1$ and merge layer M for depth $d = 2n$ to generate the candidates. In the case of the beginning depth $d = 0$, we use the initial thoughts Θ_0 as the candidates.

3.2.2 Reasonable Thought

After obtaining candidate thoughts Θ_d, our model yields reasonable thoughts Θ_d^* that constitute the final thought θ^*. In each depth d, it selects reasonable thoughts from candidate thoughts through the inference layer infer with a premise vector P_d.

Infer. The inference layer calculates the correlation score between the premise vector P_d and each candidate thought $\theta \in \Theta_d$ using multi-head attention $\mathrm{A}(Q, K = V)$ and feed-forward network FF with sigmoid function σ to evaluate if a thought is acceptable within the premises:

$$\mathrm{infer}(\mathrm{P}_d, \theta) = \sigma(\mathrm{A}(\mathrm{FF}(\theta), \mathrm{P}_d)W_r + b_r)$$
$$\text{where } W_r \in \mathbb{R}^{H \times 1}, b_r \in \mathbb{R}.$$

A thought θ is *reasonable* if its correlation score $\mathrm{infer}(\mathrm{P}_d, \theta)$ exceeds a threshold $t_r = 0.5$. In the next iteration $d + 1$, the reasonable thoughts in the current depth Θ_d^* become the input.

Update Premises. A previously obtained consequence can become a premise for the next inference step. Accordingly, our model updates the premise vector P_d with the reasonable thoughts Θ_d^* obtained in the current depth d to prepare a premise vector for the next step P_{d+1}. It gains the updated premise vector by concatenating all reasonable thoughts Θ_d^* after the multi-head attention A using the parameters of the inference layer infer:

$$\mathrm{P}_{d+1} = \mathrm{P}_d \parallel \mathrm{A}(\mathrm{FF}([\Theta_d^*]), \mathrm{P}_d).$$

Algorithm 1 Thought Expansion Process of ATHENA

Input: $\Theta_0, \mathrm{P}_0, \mathrm{G}$
Output: $\mathcal{E}^*$
  $d \leftarrow 0$
  $\Theta_0^* \leftarrow \{\theta \mid \theta \in \Theta_0, \mathrm{infer}(\mathrm{P}_0, \theta) \geq t_r\}$
  while $d \leq D$ **or** $\exists \theta \in \Theta_d^* (\mathrm{answer}(G, \theta) > t_f)$ **do**
    $\mathrm{P}_{d+1} \leftarrow \mathrm{P}_d \parallel \mathrm{A}(\mathrm{FF}([\Theta_d^*]), \mathrm{P}_d)$
    $d \leftarrow d + 1$
    if $d = 1, 3, 5 \ldots$ **then**
      $\Theta_d \leftarrow \bigcup_{\mathrm{op} \in \{-\cdot, \cdot^{-1}\}} \{\mathrm{T}^{\mathrm{op}}(\theta) \mid \theta \in \Theta_{d-1}^*\}$
    else if $d = 2, 4, 6 \ldots$ **then**
      $\Theta_d \leftarrow \bigcup_{\mathrm{op} \in \{+, \times\}} \{\mathrm{M}^{\mathrm{op}}(\theta_i, \theta_j) \mid \theta_i, \theta_j \in \Theta_{d-1}^*\}$
    end if
    $\Theta_d^* \leftarrow \Theta_{d-1}^* \cup \{\theta \mid \theta \in \Theta_d, \mathrm{infer}(\mathrm{P}_d, \theta) \geq t_r\}$
  end while
  $\theta^* \leftarrow \arg\max_{\theta \in \Theta_d^*} \mathrm{answer}(G, \theta)$
  return $\mathcal{E}(\theta^*)$

3.3 Final Thought

A final thought θ^* is the answer to the question. When the thought expansion process finishes, our model decides the final thought by selecting a thought with the maximum score. We have two criteria to terminate the iteration; (1) when the depth reaches the maximum expansion depth D; (2) if there is a thought with the score that exceeds a confidence threshold t_f on iteration. We calculate the score of each reasonable thought $\theta \in \Theta_d^*$ using the multi-head attention A and feed-forward network FF with the goal vector G, activated by sigmoid σ:

$$\mathrm{answer}(\mathrm{G}, \theta) = \sigma(\mathrm{A}(\mathrm{FF}(\theta), \mathrm{G})W_a + b_a),$$

where $W_a \in \mathbb{R}^{H\times 1}, b_a \in \mathbb{R}$.

A thought with the maximum score in the reasonable thoughts becomes a final thought θ^*:

$$\theta^* = \arg\max_{\theta \in \Theta_d^*} \left(\text{answer}(G, \theta)\right).$$

The model bestows the final thought the fidelity to shape the answer to the goal of the problem.

Algorithm 1 shows the overall process to derive the final answer $\mathcal{E}(\theta^*)$ from inputs $\Theta_0, \mathrm{P}_0, \mathrm{G}$.

4 Experiments

We conduct experiments across a comprehensive range of MWP) solving tasks to show that ATHENA outperforms strong baselines in both full datasets and variant versions of the original datasets while being more interpretable in terms of intermediate steps toward the answers.

4.1 Experimental Setups

Baselines. We select four representative approaches as the baselines to compare with ATHENA: Transformer (Vaswani et al., 2017)[2], a goal-driven tree-structured model (GTS) (Xie and Sun, 2019), Graph-to-Tree (Zhang et al., 2020b)[3] and DeductReasoner (Jie et al., 2022).[4] Transformer is a sequence-to-sequence approach that uses multi-head attention mechanism while GTS is a strong baseline of sequence-to-tree model. Graph-to-Tree is another approach that adds a graph encoder on top of GTS. We adopt DeductReasoner as an additional baseline that introduces a complex relation extraction method for deductive steps and hence achieves the state-of-the-art performance.

Implementation Details. We use RoBERTa-base and RoBERTa-large as our base pre-trained embeddings (Liu et al., 2019b) and Chinese-RoBERTa (Cui et al., 2019) for Chinese benchmarks to compare our baselines. We use pre-layer normalization (Xiong et al., 2020) for our multi-head attention method to fully leverage a dynamic range of embeddings. We set D by the maximum value of the reasoning depth of test examples for each dataset.[5]

We set $t_f = 0.95$ and train our model by giving ideal accepted prior thoughts Θ^*_{d-1} and labels of infer and answer in each depth to calculate the loss with binary cross entropy over all labels.[6] We perform our experiments with Nvidia RTX 3090 GPU.

Dataset. We test ATHENA on both standard MWP benchmarks and relatively new benchmarks that contain various linguistic expressions in contexts or questions for mathematical reasoning. The standard benchmarks are **MAWPS** (Koncel-Kedziorski et al., 2016), **ASDiv-A** (Miao et al., 2020), and **Math23k** (Wang et al., 2017). MAWPS is an English corpus collected from the online MWP repository, and Math23k is a Chinese corpus crawled from online posts. ASDiv-A is an acronym of An arithmetic subset of Academia Sinica Diverse dataset (ASDiv-A), consisting of diverse English lexical patterns.

The relatively new benchmarks either alter the standard benchmarks or vary the grounded expressions from the collected data to evaluate the model performance without bias from learned data. **SVAMP** (Patel et al., 2021) varies in the components of one of the standard benchmarks, ASDiv-A to evaluate various contextual expressions on elementary-level arithmetic problems. **UnbiasedMWP** (Yang et al., 2022) is an online-crawled Chinese corpus that augments the questions from the same context to evaluate models if they are able to generate adequate corresponding mathematical expressions. We split MAWPS, ASDiv-A, Math23k, SVAMP, following Jie et al. (2022) and Patel et al. (2021), respectively.

One-to-Many Test. In addition to the standard test, we conduct one-to-many variants tests to measure model generalization to many variant questions from one example within the common context. We select two datasets SVAMP and UnbiasedMWP to apply for this test. Each example in the dataset has a problem sequence that is composed of context and question descriptions. Within the groups by context, we split the examples one-to-many. One randomly selected example per group goes to a training set while the rest examples in the group move to a test set. We use the examples that do not have other variants within the context group as a validation set. We name the resorted SVAMP

[2]We follow hyperparameters by Lan et al. (2022) for both vanilla transformer and RoBERTa-based transformer.

[3]We follow the best hyperparameter settings in Patel et al. (2021) for both vanilla models and RoBERTa-based models.

[4]We use their hyperparameter setups. We use the MAWPS setup for testing ASDiv-A, and use the Math23k setup for UnbiasedMWP. Since the authors do not provide setups for RoBERTa-large, we optimize the model and report the best score with half batch size and half learning rate from those used in the RoBERTa-base setup.

[5]We present the values of each dataset in Table 8.

[6]We present detailed training settings and hyperparameters in Appendix A

	MAWPS	ASDiv-A	Math23k	SVAMP	UnbiasedMWP	SVAMP (1:N)	UnbiasedMWP (1:N)
Language	English	English	Chinese	English	Chinese	English	Chinese
Random embedding							
Transformer	85.6	59.3	61.5	20.7	20.5±0.73	9.7±0.19 (14.9)	16.9±0.31 (51.5)
GTS	82.6	71.4	75.6	30.8	26.2±0.20	12.2±0.37 (43.8)	22.8±0.22 (65.0)
Graph-to-Tree	83.7	77.4	77.4	36.5	27.2±0.37	25.3±0.12 (52.5)	24.3±0.25 (66.4)
RoBERTa-base							
R-Transformer	88.4	72.1	76.9	30.3	18.3±0.15	13.5±0.33 (33.4)	14.9±0.20 (53.1)
R-GTS	88.5	81.2	-	41.0	-	40.9±0.50 (64.4)	-
R-Graph-to-Tree	88.7	82.2	-	43.8	-	31.8±0.36 (66.7)	-
DeductReasoner	92.0±0.20	83.1±0.24	**85.1**±0.24	45.0±0.10	31.6±0.51	42.5±0.41 (69.1)	26.5±0.55 (79.5)
ATHENA(Ours)	**92.2**±0.10	**86.4**±0.11	84.4±0.24	**45.6**±0.50	**36.2**±0.67	**52.5**±0.50 (70.1)	**35.4**±0.45 (80.5)
RoBERTa-large							
DeductReasoner	92.6±0.16	89.1±0.46	85.8±0.42	50.3±0.30	34.9±0.11	51.6±0.38 (75.4)	33.7±0.60 (83.2)
ATHENA(Ours)	**93.0**±0.20	**91.0**±0.13	**86.5**±0.25	**54.8**±0.63	**42.0**±0.57	**67.8**±0.58 (79.8)	**48.4**±0.38 (84.8)

Table 2: Comparison of MWP methods. We use MAWPS, ASDiv-A, and Math23k for standard evaluation, SVAMP and UnbiasedMWP to evaluate the ability to solve entirely unseen, various expressions, and SVAMP and UnbiasedMWP with the one-to-many test to estimate the adaptability of confusing linguistic subtlety.

and UnbiasedMWP using the one-to-many setup as SVAMP(1:N) and UnbiasedMWP(1:N). We construct these sets with 5 different random seeds to mitigate training bias and report the average performance.

4.2 Results

We repeat our experiments 5 times with different random seeds and report the average answer accuracy with the standard error. We report results on multiple benchmarks, variants splitting tests, the impact of pre-trained language models depending on their size, and ablation tests.

Overall Performance. Table 2 shows the performance of different methods on 7 benchmarks. ATHENA establishes new state-of-the-art results for overall benchmarks. ATHENA outperforms prior MWP methods on all occasions with one exception of its performance on Math23k when trained on the RoBERTa-base model. When compared to the most competitive work DeductReasoner, our model obtains a relative improvement of 3.84%p on total benchmarks.

Performance on One-to-Many Test. We note that ATHENA achieves large performance gains compared to the second-best method, from 42.5% to 52.4% and from 26.5% to 35.0% on SVAMP (1:N) and UnbiasedMWP (1:N), respectively. As illustrated in Section 4.1, we evaluate our model on SVAMP (1:N) by training with one example per problem set to test how well ATHENA reasons on the questions that use the same textual descriptions but ask for different target answers. We observe

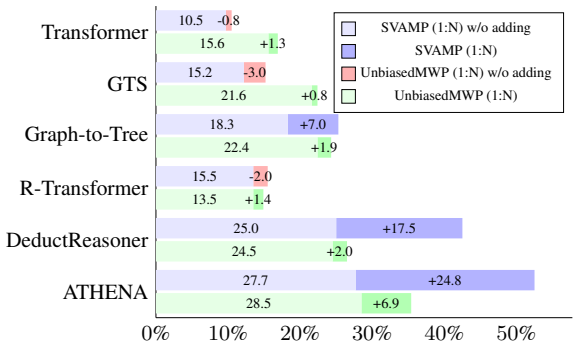

Figure 3: Accuracy changes when adding one example per context into the training set by applying the one-to-many test.

from Figure 3 that the results show ATHENA is strong at applying mathematical reasoning that is formed by unlearned patterns once the model has learned the context. Our approach is distinguished from other baselines including RoBERTa-GTS and DeductReasoner which show the opposite phenomena. Other baselines are relatively stronger on original benchmarks than on the benchmark variants including those with the one-to-many Test. Hence we reach the conclusion that ATHENA has the superiority of acknowledging the subtlety of contextual information governed by the required mathematical operations.

Dependence on Training Set. We observe that ATHENA performs well on datasets that apply the one-to-many test because our model has a sense of subtlety in terms of distinct question concepts, not because our model is reluctant to follow learned expressions. Figure 5 illustrates where the wrong prediction for the question variant experi-

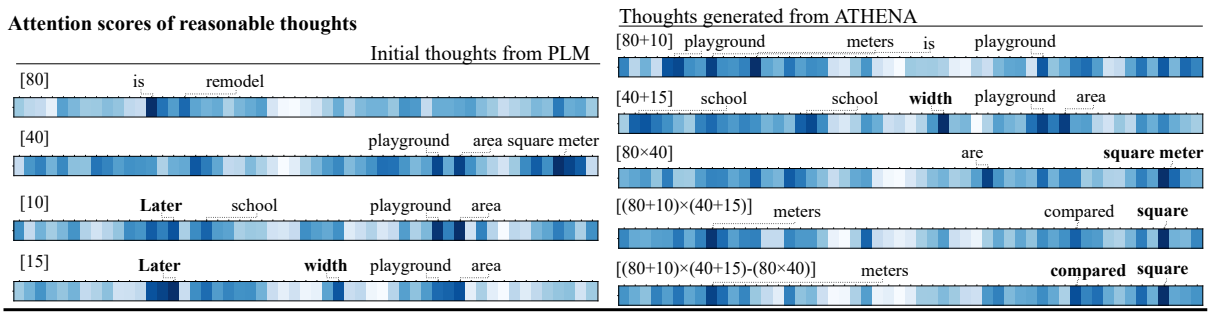

Figure 4: Visualization of reasonable thoughts from ATHENA with calculating attention score of the tokens in the problem sequence on RoBERTa-large.

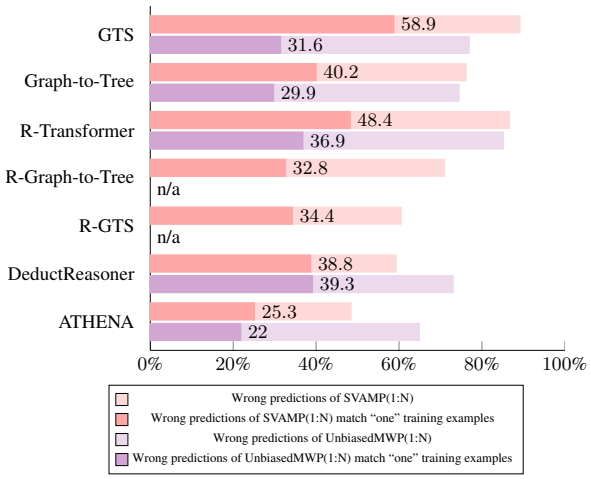

Figure 5: Percentages of the wrong predictions that match with the answers of "one" training examples in one-to-many split. The less the percentage scores, the less the method unnecessarily leans on the training bias.

ments comes from. If a model outputs equations that are labeled for questions with shared contexts when being trained, this indicates that the model relies on training data points, especially on context contents regardless of different question expressions. The result shows that our model also has the least accuracy for a golden training example. It is notable that ATHENA has the lowest score for following the trained expressions while DeductiveReasoner predicts the highest scores among other baselines that use RobBERTa, even higher than those of R-GTS or R-Graph-to-Tree on UnbiasedSVAMP(1:N). This shows that while DeudctiveReasoner can learn to solve mathematical problems, it also easily falls into learning shortcuts.

Different Sizes of PLMs. We estimate the baselines both on RoBERTa-base and RoBERTa-large models to examine the influence of the model sizes. As expected, Table 2 shows that the bigger the model size is for the embedding, the better the model performance reaches. When we estimate the accuracy gaps by increasing the model size, ATHENA achieves relatively better performance gains (7.26%p) on average for the entire benchmarks than DeductReasoner does (4.6%p). We can observe that on dataset variants, ATHENA obtains relatively more benefits from bigger model sizes (14.15%p) than DeductReasoner does (8.15%p), while both are still taking great advantage of the rich model parameters to understand the question better and to solve those confusing questions. It also shows that DeductReasoner fails to improve performance on question variants from the original datasets leveraging the additional training sets in large-scale PLM. In short, our model leverages large-scale PLM much more efficiently than the competitive model.

Visualization of Thoughts. We interpret the thoughts using attention scores between reasonable thoughts and the problem sequence.[7] As illustrated in Figure 4, we observe how the thought relates to the words. Most of the initial thoughts are related to the "playground", while the thoughts carrying the meaning of increased size show a strong correlation to the word "Later". The thoughts carrying width sizes [15] and [40+15] show high attention scores on "width", while the other thoughts do not have high attention scores on them. Thoughts that calculate the area produce high attention scores on words "square meter" or "area". The final thought

[7]We use answer layer to calculate the attention score, giving the problem sequence embedding as an input, instead of the goal vector.

	MAWPS	ASDiv-A	SVAMP	Math23k	UnbiasedMWP	SVAMP (1:N)	UnbiasedMWP (1:N)	Average
Avg depth	3.87	3.46	3.47	5.18	4.44	3.47	4.44	4.05
ATHENA	**92.2**	**86.4**	**45.6**	**85.1**	36.2	**52.5**	**35.4**	**62.0**
− update	92.1	84.8	44.9	82.7	34.9	52.4	34.7	60.9
− premise	90.6	85.0	44.7	65.7	**36.3**	51.5	34.6	58.3

Table 3: Ablation studies on premise vector construction. (1) "− update" is the premise vector without updating strategies and (2) "− premise" is the direct classification method without premise vectors.

marks a high score on "compared", which asks for the difference between the increased and original areas.

Ablation on Premise Vectors. A premise vector is a criterion for determining thoughts in each inference step to obtain the valid pathways to reach the goal. We conduct an ablation study to evaluate how ATHENA composes the premise vectors to ultimately generate optimal final thoughts.

For evaluating the impact of the premise vectors in generating reasonable thoughts, we adopt two different settings: (1) We do not *update* premise vectors but use the initial premise vector (i.e., [CLS] token) in all expansion depths: $\mathrm{P}_d = \mathrm{P}_0$. We aim to see how the existence of thoughts that update the premise vector impacts models to help find solid reasonable thoughts. (2) We do not use the premise vector and directly classify the thoughts for the next iteration: $\mathrm{infer}(\theta) = \sigma(\theta W_r + b_r)$.

Table 3 shows the results of the different premise construction strategies for reaching the appropriate conclusion. Despite slight fluctuations across different methods, ATHENA without premise vectors decreases the overall performances by up to 3.7%p compared to our proposed method. When the model does not update the premise vectors in the thought expansion iteration while still adopting the initial one, the performance decreases relatively by 1.1%p. It is notable that Math23k, a dataset of the deepest average depth, shows the performance degradation even worse, 2.4%p and 19.4%p respectively. From these observations, we conclude that the premise vector plays an important role in properly deriving the final thoughts. Especially, considering that the model applies the update on every expansion depth, the large performance gap for Math23k strongly supports our premise update method for its effectiveness.

5 Conclusion

We state that an ideal MWP model needs to be practical in real-world settings that are critical to capture the diverse applications of the same mathematical operations. For this reason, we conclude that ATHENA with thought expansion reaches significant improvements toward the ideal model due to its decent performance on unseen problems or restricted examples to learn.

Limitations

The paper has the following limitations. First, we only consider arithmetic problems, not algebraic, calculus, or other topics of mathematical problems. Especially, for a fair comparison with other models, we only evaluate the performance using MWP datasets with single equations, while the model is able to handle multi-equation problems by simply adding "=" operation on Merge. Second, we do not compare ATHENA with large-scale language models (LLMs) since we focus on acquiring knowledge from limited mathematical samples.

Ethics Statement

This work breaks down a process of reasoning from the human cognitive perspective and instantiates individual thoughts with symbolic representation so that it can clarify and handle the intermediate procedures of the model. Although the perspective may have the potential to filter harmful or toxic thoughts from the broad sight of thoughts, this work does not consider or validate the effectiveness of such applications. Therefore, we do not suggest using our work for this purpose without thorough experiments for its possibilities.

Acknowledgements

This research was supported by the NRF grant (RS-2023-00208094) and the AI Graduate School Program (No. 2020-0-01361) funded by the Korean government (MSIT). Han is a corresponding author.

References

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. *arXiv preprint math/0701393*.

James P Byrnes and Barbara A Wasik. 1991. Role of conceptual knowledge in mathematical procedural learning. *Developmental psychology*, 27(5):777.

Katherine H Canobi. 2009. Concept–procedure interactions in children's addition and subtraction. *Journal of experimental child psychology*, 102(2):131–149.

Ting-Rui Chiang and Yun-Nung Chen. 2019. Semantically-aligned equation generation for solving and reasoning math word problems. 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 2656–2668.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*.

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

Zhenya Huang, Xin Lin, Hao Wang, Qi Liu, Enhong Chen, Jianhui Ma, Yu Su, and Wei Tong. 2021. Disenqnet: Disentangled representation learning for educational questions. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, page 696–704.

Zhenya Huang, Qi Liu, Weibo Gao, Jinze Wu, Yu Yin, Hao Wang, and Enhong Chen. 2020. Neural mathematical solver with enhanced formula structure. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 1729–1732.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Zhanming Jie, Jierui Li, and Wei Lu. 2022. Learning to reason deductively: Math word problem solving as complex relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5944–5955.

Philip Johnson-Laird. 2008. *How we reason*. Oxford University Press.

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

Yihuai Lan, Lei Wang, Qiyuan Zhang, Yunshi Lan, Bing Tian Dai, Yan Wang, Dongxiang Zhang, and Ee-Peng Lim. 2022. Mwptoolkit: An open-source framework for deep learning-based math word problem solvers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11):13188–13190.

Jierui Li, Lei Wang, Jipeng Zhang, Yan Wang, Bing Tian Dai, and Dongxiang Zhang. 2019. Modeling intra-relation in math word problems with different functional multi-head attentions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6162–6167.

Shucheng Li, Lingfei Wu, Shiwei Feng, Fangli Xu, Fengyuan Xu, and Sheng Zhong. 2020. Graph-to-tree neural networks for learning structured input-output translation with applications to semantic parsing and math word problem. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2841–2852.

Zhongli Li, Wenxuan Zhang, Chao Yan, Qingyu Zhou, Chao Li, Hongzhi Liu, and Yunbo Cao. 2022. Seeking patterns, not just memorizing procedures: Contrastive learning for solving math word problems. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2486–2496.

Zhenwen Liang, Jipeng Zhang, Lei Wang, Wei Qin, Yunshi Lan, Jie Shao, and Xiangliang Zhang. 2022. MWP-BERT: Numeracy-augmented pre-training for math word problem solving. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 997–1009.

Xin Lin, Zhenya Huang, Hongke Zhao, Enhong Chen, Qi Liu, Hao Wang, and Shijin Wang. 2021. Hms: A hierarchical solver with dependency-enhanced understanding for math word problem. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4232–4240.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. 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 (Volume 1: Long Papers)*, pages 158–167.

Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. 2019a. Tree-structured decoding for solving math word problems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International*

Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2370–2379.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

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

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2144–2153.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094.

Jinghui Qin, Lihui Lin, Xiaodan Liang, Rumin Zhang, and Liang Lin. 2020. Semantically-aligned universal tree-structured solver for math word problems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3780–3789.

Bethany Rittle-Johnson and Martha Wagner Alibali. 1999. Conceptual and procedural knowledge of mathematics: Does one lead to the other? *Journal of educational psychology*, 91(1):175.

Bethany Rittle-Johnson and Michael Schneider. 2014. Developing conceptual and procedural knowledge of mathematics. In *The Oxford Handbook of Numerical Cognition*. Oxford University Press.

Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. 2021. Generate & rank: A multi-task framework for math word problems. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2269–2279.

Yibin Shen and Cheqing Jin. 2020. Solving math word problems with multi-encoders and multi-decoders. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2924–2934.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018. Translating a math word problem to a expression tree. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1064–1069.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Qinzhuo Wu, Qi Zhang, and Zhongyu Wei. 2021. An edge-enhanced hierarchical graph-to-tree network for math word problem solving. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1473–1482.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5299–5305.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 10524–10533.

Zhicheng Yang, Jinghui Qin, Jiaqi Chen, and Xiaodan Liang. 2022. Unbiased math word problems benchmark for mitigating solving bias. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1401–1408.

Weijiang Yu, Yingpeng Wen, Fudan Zheng, and Nong Xiao. 2021. Improving math word problems with pre-trained knowledge and hierarchical reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3384–3394.

Jipeng Zhang, Roy Ka-Wei Lee, Ee-Peng Lim, Wei Qin, Lei Wang, Jie Shao, and Qianru Sun. 2020a. Teacher-student networks with multiple decoders for solving math word problem. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4011–4017.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020b. Graph-to-tree learning for solving math word problems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3928–3937.

Appendices

A Training Details

In this section, we provide detailed information about our training settings.

Loss. Given an answer equation $\mathcal{E}$, let $\mathbb{1}_{\text{infer}}(\theta)$ denote the target of infer for a thought θ and $\mathbb{1}_{\text{answer}}(\theta)$ denote the target of the final decision answer for a thought θ:

$$\mathbb{1}_{\text{infer}}(\theta) = \mathbb{1}(\mathcal{E}(\theta) \subseteq \mathcal{E}),$$
$$\mathbb{1}_{\text{answer}}(\theta) = \mathbb{1}(\mathcal{E}(\theta) \equiv \mathcal{E}),$$

where $\mathcal{E}(\theta) \subseteq \mathcal{E}$ denotes that $\mathcal{E}$ contains the sub-expression $\mathcal{E}(\theta)$ (e.g., "$(a+b)$" $\subseteq$ "$(a+b)\times c$").

Let BCE denote the binary cross entropy function, the training objective is to minimize loss $\mathcal{L}$:

$$\mathcal{L} = \frac{1}{|\bigcup_d \Theta_d| + |\Theta^*_d|}\Big(\sum_{\theta \in \bigcup_d \Theta_d} BCE(\text{infer}(\theta), \mathbb{1}_{\text{infer}}(\theta)) + \sum_{\theta \in \Theta^*_d} BCE(\text{answer}(\theta), \mathbb{1}_{\text{answer}}(\theta))\Big).$$

Optimizer. We use AdamW optimizer (Loshchilov and Hutter, 2017) with weight decay $\omega = 10^{-5}$. Learning rate lr_e for each epoch e is decayed every S_{lr} epoch with factor γ starting from lr:

$$lr_e = lr \cdot \gamma^{[e/S_{lr}]}.$$

Regularization. We adopt dropout with probability p to every layer and stochastic weight averaging (Izmailov et al., 2018) for last $epoch_{swa}$ epochs.

Hyperparameters. We present our experiments for hyperparameters in Table 4, with the bold text denoting the best performance. We train our model for 100 epochs. In the result, we observe that RoBERTa-base and RoBERTa-large share the best hyperparameter settings except for learning rate lr.

B Dataset Statistics

In this section, we show the statistics of datasets and their requirements.

One-to-many Split. In Section 4.1, we explain building one-to-many dataset splits. We provide how many groups and examples are made from the contexts in Table 5.

Number of Thoughts. We present the required number of thoughts for each dataset in Table 6. While Math23k requires a large number of candidate thoughts in total depth, we show a thought expansion in each depth does not require huge memory space. Therefore, efficient implementation strategies such as removing unselected candidate thoughts from memory space are enough to manage computational resources.

C Additional Experiments

This section presents additional studies to further clarify the robustness and fairness of our experiments for some minor strategies by showing their performance independence.

Punctuation Mark. In Section 3, we initialize goal vector G with the punctuation mark of the question sequence or the last punctuation mark (i.e., the question mark in most cases). The motivation of this strategy is from Clark et al. (2019) showing the punctuation mark gets high attention from other tokens in the last layers. Intuitively, high attention can generalize the question sequence, so we conduct experiments to evaluate the generalization ability of the punctuation mark compared to using all question sequences as a goal vector G. We conduct experiments for all datasets except Math23k (Wang et al., 2017) since it does not provide the question subsequence.

As shown in Table 7, using the punctuation mark effectively generalizes the question to represent a goal in most cases. It shows even better performances than using the question sequence. From an intuitive interpretation, the question sequence holds some tokens that are not informative for reasoning targets, so a punctuation mark representation helps the model to focus on a reasoning goal.

Stop Criteria. In Section 3.3 and Algorithm 1, we present the two stop criteria: (1) the depth reaches the maximum expansion depth D, or (2) one of the final scores exceeds a threshold t_f. In addition to the main experiments setting the D and t_f with an arbitrary value, we conduct the experiments of the higher maximum expansion depth and $t_f = 0.5$ to show the performance differences from the values. As shown in Table 8, the scores are fairly equal with a trivial gap. This demonstrates that the performances of our model do not rely on stop criteria parameters but are solid achievements.

	Batch Size	lr	S_{lr}	γ	p	$epoch_{swa}$
RoBERTa-base	[**4**, 8]	[5e-6, 7e-6, 1e-5, **1.3e-5**, 1.5e-5, 2e-5]	[**10**, 15, 20]	[**0.5**, 0.7]	[0.1, **0.5**]	[**30**, 50, 70]
RoBERTa-large	[**4**, 8]	[5e-6, **7e-6**, 1e-5, 1.3e-5, 1.5e-5, 2e-5]	[**10**, 15, 20]	[**0.5**, 0.7]	[0.1, **0.5**]	[**30**, 50, 70]

Table 4: Hyperparameter search spaces of ATHENA

	SVAMP (1:N)	**UnbiasedMWP (1:N)**
# examples in original split	3138 / 0 / 1000	2507 / 200 / 685
# groups of single examples	438	45
# groups of multiple examples	205	154
# examples in one-to-many split	3343 (+205) / 438 (+438) / 357 (-562)	2661 (+154) / 245 (+45) / 486 (-199)

Table 5: Statistics of one-to-many test splits

Dataset	# candidates in total depth			# in a reasoning path			# candidates in last depth			depth of reasoning path		
	min	average	max	min	average	max	min	average	max	min	average	max
MAWPS	17	45.40±0.46	192	2	4.52±0.03	12	4	9.49±0.08	48	2	3.87±0.03	11
ASDiv-A	16	26.86±0.42	71	3	4.10±0.03	7	6	9.65±0.09	22	1	3.46±0.02	5
SVAMP	2	28.09±0.44	70	1	4.23±0.03	7	2	10.54±0.10	22	1	3.47±0.03	5
Math23k	4	65.1±0.31	939	1	6.33±0.02	29	2	14.85±0.06	108	1	5.18±0.01	41
U.MWP	5	47.0±0.47	214	1	5.18±0.03	13	2	11.67±0.11	48	1	4.44±0.02	11

Table 6: Statistics of thoughts that are required for each dataset

	MAWPS	**ASDiv-A**	**SVAMP**	**UnbiasedMWP**	**SVAMP (1:N)**	**UnbiasedMWP (1:N)**	**Average**
Avg depth	3.87	3.46	3.47	4.44	3.47	4.44	4.05
RoBERTa-base							
punctuation mark	**92.2**	**86.4**	**45.6**	36.2	**52.5**	**35.4**	**58.1**
question sequence	92.0	86.3	44.9	**36.3**	51.0	33.4	57.3
RoBERTa-large							
punctuation mark	**93.0**	91.0	**54.8**	**42.0**	**67.8**	**48.4**	**66.2**
question sequence	92.9	**91.2**	54.4	41.0	66.9	46.8	65.5

Table 7: Comparing goal vector using the whole question sequence from the punctuation mark

Stop Criteria	**MAWPS** D=7	**ASDiv-A** D=5	**SVAMP** D=5	**Math23k** D=19	**UnbiasedMWP** D=9	**Average**
$t_f = 0.95$, max. expansion depth=D	**92.2±0.10**	86.4±0.11	**45.6±0.50**	84.4±0.24	36.2±0.67	**69.0**
$t_f = 0.5$, max. expansion depth=D	92.0±0.15	86.3±0.24	45.3±0.37	84.7±0.20	36.0±0.39	68.9
$t_f = 0.95$, max. expansion depth=$D + 2$	91.9±0.07	**86.5±0.28**	45.2±0.41	84.6±0.18	35.8±0.75	68.8
$t_f = 0.95$, max. expansion depth=$D + 4$	92.0±0.09	**86.5±0.28**	45.2±0.41	**84.8±0.27**	**36.4±0.44**	**69.0**

Table 8: Performances among the different parameters of stop criteria