Knowledge-Centered Dual-Process Reasoning for Math Word Problems With Large Language Models

Jiayu Liu, Zhenya Huang, *Member, IEEE*, Qi Liu, *Member, IEEE*, Zhiyuan Ma, Chengxiang Zhai, and Enhong Chen, *Fellow, IEEE*

***Abstract*—Math word problem (MWP) serves as a critical milestone for assessing the text mining ability and knowledge mastery level of models. Recent advancements have witnessed large language models (LLMs) showcasing remarkable performance on MWP. However, current LLMs still frequently exhibit logical errors, which highlights their inability to fully grasp the knowledge required for genuine step-by-step mathematical reasoning. To this end, in this paper, we propose a novel Knowledge-guided Solver (KNOS) framework that empowers LLMs to simulate human mathematical reasoning, whose core idea is to *Invoke-Verify-Inject* necessary knowledge to solve MWP. We draw inspiration from the dual-process theory to construct two cooperative systems: a *Knowledge System* and an *Inference System*. Specifically, the *Knowledge System* employs LLMs as the knowledge base and develops a novel *knowledge invoker* that can elicit their relevant knowledge to support the strict step-level mathematical reasoning. In the *Inference System*, we propose a *knowledge verifier* and a *knowledge injector* to evaluate the knowledge rationality and further guide the step-wise symbolic deduction in an interpretable manner based on human cognitive mechanism, respectively. Moreover, to tackle the potential scarcity issue of mathematics-specific knowledge in LLMs, we consider an open-book exam scenario and propose an improved version of KNOS called EKNOS. In EKNOS, we meticulously design *knowledge selectors* to extract the most relevant commonsense and math formulas from external knowledge sources for each reasoning step. This knowledge is utilized to assist the *knowledge invoker* in better stimulating LLMs' reasoning abilities. Both KNOS and EKNOS are flexible to empower different LLMs. Our experiments with GPT3, ChatGPT, and GPT4 not only demonstrate their reasoning accuracy improvement but also show how they bring the strict step-wise interpretability of mathematical thinking.**

***Index Terms*—Knowledge reasoning, math word problem, large language model.**

Received 14 June 2024; revised 7 March 2025; accepted 22 March 2025. Date of publication 1 April 2025; date of current version 1 May 2025. This research was partially supported in part by the National Key Research and Development Program of China under Grant 2021YFF0901005, in part by the Key Technologies R&D Program of Anhui Province under Grant 202423k09020039, in part by the National Natural Science Foundation of China under Grant 62477044 and Grant 62337001, and in part by the Fundamental Research Funds for the Central Universities under Grant WK2150110038. The work of Zhenya Huang was supported by the Young Elite Scientists Sponsorship Program by CAST under Grant 2024QNRC001. Recommended for acceptance by W. Ding. *(Corresponding author: Zhenya Huang.)*

Jiayu Liu and Qi Liu are with the School of Data Science, State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China, Hefei 230000, China (e-mail: jy251198@mail.ustc.edu.cn; qiliuql@ustc.edu.cn).

Zhenya Huang, Zhiyuan Ma, and Enhong Chen are with the School of Computer Science and Technology, State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China, Hefei 230000, China (e-mail: huangzhy@ustc.edu.cn; zhyma@mail.ustc.edu.cn; cheneh@ustc.edu.cn).

Chengxiang Zhai is with the Department of Computer Science, University of Illinois at Urbana-Champaign, Champaign, IL 61820 USA (e-mail: czhai@illinois.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TKDE.2025.3556367, provided by the authors.

Digital Object Identifier 10.1109/TKDE.2025.3556367

I. Introduction

MATHEMATICAL problems represent a highly abstract and symbolic type of data. Automatically solving these problems requires models to thoroughly mine information from the problem text and possess the necessary mathematical knowledge, which therefore has garnered increasing attention in the field of data mining [1]. Specifically, researchers have developed various tasks for different types of mathematical problems, such as math word problem, geometry problem, theorem proving, etc. [2], [3], [4], [5]. Especially, the recent large language models (LLMs) such as GPT3 [6], ChatGPT,[1] and GPT4 [7] have demonstrated a noteworthy level of mathematical problem-solving ability [8]. Elicited by methods such as Chain-of-Thought (CoT) [9], they also exhibit potential to perform intricate and step-by-step reasoning.

Nevertheless, behind these successes, numerous studies have indicated that the current LLMs are still prone to exhibit logical errors when solving problems [10], [11], [12], [13]. They often generate content that is not grounded in the context of arithmetics, even within a reasoning chain [14], [15]. For instance, on widely-used benchmarks like MATH [16], MathBench [17], and SciBench [18], GPT4 and ChatGPT consistently fall below 80% and 50% accuracy, respectively. Similarly, on the challenging STREET benchmark [12] that demands structured, step-by-step reasoning for problems spanning over 15 steps, the accuracy of GPT3's logical graph remains under 20%. Our experiments in Section VI-B also observe error rates from 5.9% to 81.9% for GPT3, ChatGPT, and GPT4 on Math23K [2] and MAWPS [19] datasets. In Fig. 1, we take the basic math word problem (MWP) [1] as a more concrete example. This problem requires a model to read the sentence "Allyson plays... one season?" and then derive a mathematical formal expression "$2 \times 10 \times (10 - 1) \div 2$" to get the answer "90". However, even for such a simple problem, ChatGPT, a sophisticated model with 175 billion parameters, responds with an erroneous solution "Each team... $\boxed{180}$". This propensity for mistakes on problems at such an elementary school level raises legitimate concerns:

[1] https://openai.com/blog/chatgpt/.

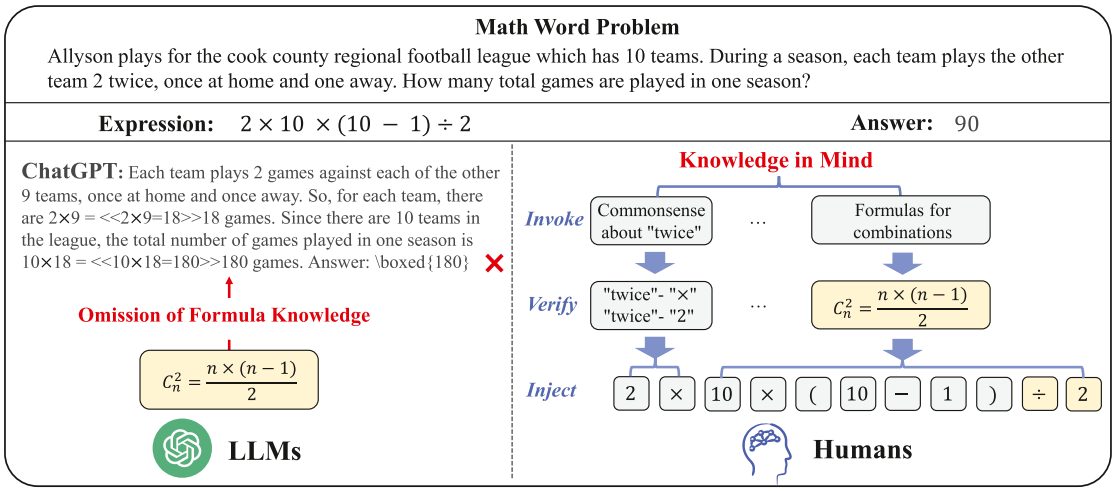

Fig. 1. Example of LLMs' and human mathematical reasoning process on Math Word Problem (MWP).

Have LLMs genuinely mastered mathematical reasoning ability [8]?

The cause for such errors is that the problem-solving process of current LLMs is not aligned with the knowledge-centered reasoning process of humans. Specifically, epistemological theories [20], [21] have revealed that humans generally perform step-wise reasoning. This process is essentially condensed into the formation of abstract mathematical expressions, where each step relies on necessary *knowledge* to guide logical and grounded thinking. Take Fig. 1 as an example, when reasoning the symbol "÷", humans may first *invoke* essential knowledge (e.g., formulas for calculating combinations) in their mind. Then, they *verify* the rationality of the invoked knowledge and keep the relevant ones (e.g., formula "$C_n^2 = \frac{n \times (n-1)}{2}$" [22]). Based on this, they ultimately *inject* the knowledge into specific symbol derivation ("÷2"). In comparison, as indicated in the literature, although CoT could help guide LLMs through multiple steps, LLMs still work as a black box, whose knowledge and corresponding application are implicitly mixed in the continuous token prediction [23]. This process lacks control and thus suffers from the risks of errors and defects (e.g., ChatGPT lacks the utilization of the formula in Fig. 1).

The reasoning errors of LLMs, coupled with their inability to explain their answers, undermine their trustworthiness and hinder their applications. To address this limitation of current LLMs and potentially enhance their reasoning capacity, in this paper, we aim to investigate two research questions: *Q1.) Have LLMs mastered mathematical knowledge?* and *Q2.) Can LLMs adequately apply their knowledge to perform human-like reasoning?* To answer these questions, we explore the potential of utilizing LLMs to simulate the cognitive process of how humans *Invoke-Verify-Inject* knowledge for conducting mathematical reasoning. However, many challenges remain to be tackled. First, reasoning on different steps may require different knowledge. For example, in Fig. 1, reasoning "2×" requires the commonsense knowledge of "twice", but reasoning "÷" requires math formula "$C_n^2 = \frac{n \times (n-1)}{2}$". It is challenging to mimic this precise step-wise knowledge invocation for LLMs. Second, the elicitation of LLMs' implicit knowledge may be very prone to hallucination [8], [15]. Therefore, the evaluation of the quality of invoked knowledge is necessary, yet it remains underexplored. Third, for knowledge injection, the reasoning process of LLMs is continuous natural language generation in essence, which is different from genuine reasoning steps in the mathematical sense (even for CoT). Consequently, the knowledge is hard to guide LLMs' reasoning pathways to exert its effect in full efficiency.

To address the above challenges, in this paper, we propose a novel Knowledge-guided Solver (KNOS) framework that employs LLMs to realize the human reasoning process for MWP solving. We draw insight from the dual process theory [24], [25] that elaborates human cognitive structure to construct two cooperative systems: *Knowledge System* and *Inference System*. *Knowledge System* plays the role of human brain that stores and provides knowledge, and the *Inference System* analytically applies this knowledge to reason the answer symbol by symbol. Specifically, in *Knowledge System*, we utilize LLMs as knowledge bases and design a *knowledge invoker* to elicit their knowledge at strict mathematical step level, which can provide the most relevant knowledge for each step. In *Inference System*, we first devise a *knowledge verifier* to evaluate knowledge rationality according to intermediate reasoning states. Then, we develop a *knowledge injector* inspired by dual mechanism of human cognition [26] to integrate it into specific symbol derivation.

KNOS relies entirely on the knowledge within the large models themselves, analogous to a student taking a closed-book exam. Moreover, considering that a LLM's mathematical knowledge may be limited by its own training data (e.g., GPT3 and ChatGPT may has not learned math-specific knowledge like combination formula in Fig. 1), we further extend KNOS to EKNOS that simulates an open-book scenario of student taking exams in real life. Specifically, we offer two types of external knowledge sources including commonsense knowledge and math formulas in *Knowledge System*. For these knowledge, we propose elaborate knowledge selectors to extract the most relevant ones during each reasoning step. Then we incorporate them in *knowledge invoker* to elicit more comprehensive knowledge of LLMs, consequently enhancing the validity of each step.

Our KNOS and EKNOS are general frameworks that can empower different LLMs. Experiments with the most representative GPT3, ChatGPT, and GPT4 on two datasets verify their improvements on answer accuracy and the effectiveness in mastering knowledge for mathematical reasoning. The contributions of this paper are:

- We draw insight from dual process theory to propose a Knowledge-guided Solver (KNOS) framework and an extended EKNOS framework, both of which are general to empower existing LLMs with human-like knowledge application for conducting reliable mathematical reasoning.
- We design novel methods to *invoke*, *verify*, and *inject* knowledge of LLMs and leverage *open-book external knowledge* at step-level in strict mathematical sense, which can more effectively exert the guidance of knowledge to LLMs' reasoning process.
- We conduct experiments with three representative LLMs and compare with current LLM-enhanced methods. The results clearly demonstrate the effectiveness and superiority of our frameworks.

This paper is organized as follows. In Section II, we review existing works related to Math Word Problem (MWP) and Large Language Models (LLM). Then, we give the problem definition in Section III. In Section IV, we introduce our KNOS framework, detailing its two main structures: *Knowledge System* and *Inference System* in Sections IV-A and IV-B, respectively. In Section IV-A, we describe the *knowledge invoker*, and in Section IV-B, we elaborate on the *knowledge verifier* and *injector*. Then, in Section V, we present the extended EKNOS framework. Finally, we conduct experiments to validate our frameworks in Section VI, discuss future research and conclude the paper in Section VII.

II. Related Work

Math Word Problem: Mathematical problem solving is a key data mining task in exploring how models can acquire human-like text understanding, knowledge discovery, and logical reasoning from data [1]. Among these, math word problem (MWP) is a critical and fundamental branch, which has received extensive attention since the 1960s [27]. Due to the well-defined logic behind MWP, it provides excellent opportunities for studying and improving the reasoning capacity of LLMs, which we leverage in our work. Early work on MWP mainly focused on using human-designed rules, templates, and programs [28]. In recent years, inspired by language translation, methods using deep neural networks have been continuously developed. For example, Wang et al. [2] first proposed DNS to directly translate problems into mathematical expressions. Based on this, subsequent work has focused partly on enriching the model's semantic knowledge [29], [30], [31] and partly on mastering knowledge of specific reasoning patterns [32], [33], [34]. In the former, Zhang et al. [29] proposed Graph2Tree to strengthen understanding of relationships between numbers and words, while Wu et al. [30] introduced concrete encoding of the numbers. Besides, pretrained language models including BERT [31], RoBERTA [35] and BART [36] have also been used to lay a semantic foundation for reasoning. In the latter, Xie et al. [32] proposed Seq2Tree that grasped a top-down decomposition reasoning pattern, while Jie et al. [35] proposed DEDUCT-REASONER that inferred and combined sub-expressions in a bottom-up manner. Besides, Liu et al. [22] and Yang et al. [37] introduced math formulas, which devised elaborate formula-guided reasoning mechanisms and formula prompt for problems, respectively. Concerning LLMs, research mainly focused on how to use them (e.g., GPT-J) to generate natural language descriptions of the reasoning process [38], reasoning paths [39], or to generate a piece of code to solve the problem [40].

Large Language Models: Recently, large language models (LLMs) like ChatGPT have made significant progress in various data mining tasks [8], [41], [42], and there has been a lot of work on enhancing their mathematical reasoning ability. Wei et al. [9] first introduced Chain-of-Thought (CoT), which requires LLMs to deduce answers step by step. Kojima et al. [43] discovered that a simple prompt "Let's think step by step" can be used in CoT to further improve accuracy and interpretability. Subsequent researchers developed diverse structures to organize the reasoning paths of large models, such as tree [44], graph [45], and program [46]. For decoding strategies, Wang et al. [47] proposed Self-Consistency to reduce the randomness of greedy decoding by majority voting. Zhang et al. [48] proposed Cumulative Reasoning, which decomposed tasks into smaller components to conduct collaborative reasoning. Besides, many works improved LLMs' reasoning capabilities from data and fine-tuning. For example, Minerva [49] is an improved version of Google's PaLM by fine-tuning on mathematical formula data, while Lightman et al. [50] designed the Process-Supervised Model that is optimized from the reasoning steps, not just the final answer. To address the issue of lacking up-to-date knowledge and further enhance LLMs' reasoning ability, a recent technique involved using Retrieval-Augmented Generation (RAG) [51]. This method supplemented LLMs by retrieving relevant documents from external information resources. Future advancements in this technique focused either on designing different retrieval methods, such as using sparse [52], [53] or dense [54], [55] retrieval, or on employing various strategies to integrate the retrieved content into the model's generation process, such as integrating at LLMs' input layer [53], output layer [56], [57], or intermediate layers [58].

Our work improves previous studies from the following three aspects. First, to the best of our knowledge, we are the first that draw insights from dual process theory to make LLMs simulate human knowledge application process for solving math word problems, where we explore and prove that explicit introduction of knowledge is essential for improving the reasoning ability of LLMs. Second, compared with the recent work [39] that is also inspired by dual process theory, our frameworks focus on explicitly invoking, verifying, and injecting knowledge into LLMs' reasoning process, which are more in line with human cognition and can reduce hallucinations. Besides, our frameworks are general enough to empower different LLMs without finetuning, and thus has better generality and efficiency. Last but not least, compared with existing methods that infuse knowledge into LLMs (e.g., RAG), our frameworks inject knowledge at

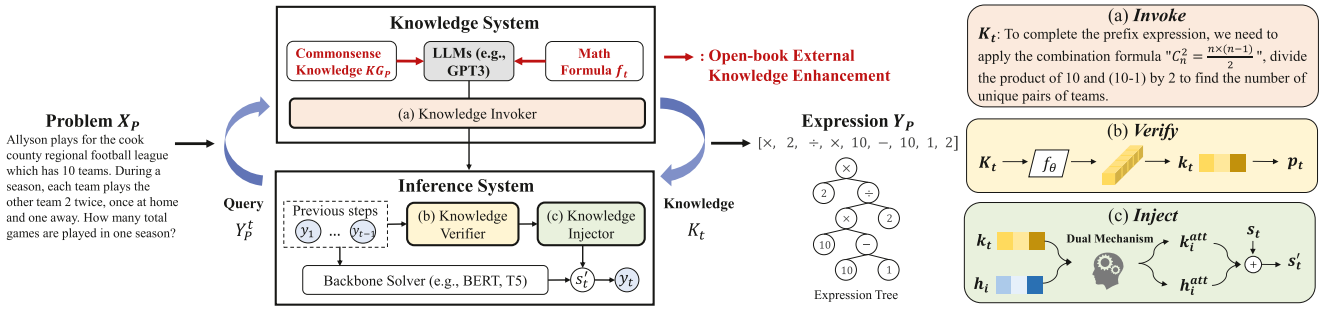

Fig. 2. Overview of our KNOS and EKNOS frameworks. The red line indicates external knowledge enhancement in EKNOS.

TABLE I
MAIN MATHEMATICAL NOTATIONS

Notation	Description
$X_P = [x_1, x_2, ..., x_n]$	input sentence of problem P
$Y_P = [y_1, y_2, ..., y_m]$	target expression of problem P
a_P	numeric answer of problem P
$Y_P^t = [y_1, y_2, ..., y_{t-1}]$	partial expression already generated at step t
$K_t = [k_t^1, k_t^2, ..., k_t^r]$	knowledge invoked at step t
$KG = (V, E)$	external commonsense knowledge graph
$KG_P = (V_P, E_P)$	knowledge graph for problem P from KG
F	external math formulas
f_t	math formula selected for step t

single symbol-level step with better mathematical significance, so our knowledge can control LLMs' reasoning process more accurately and effectively.

III. Problem Definition

Table I summarizes the main notations used in this paper. Formally, the input of a math word problem P is a sequence of n tokens: $X_P = [x_1, x_2, \ldots, x_n]$, where x_i is either a word (e.g., "Allyson", "plays" in Fig. 2) or a number (e.g., "10", "2"). The output is an expression of m symbols: $Y_P = [y_1, y_2, \ldots, y_m]$ that can be calculated to obtain a number a_P (e.g., "90") as the final answer.

Especially, to avoid the redundancy of parentheses and to ensure that the expression is computable, most current work targets at generating an expression tree (Fig. 2) and outputs its prefix expression as Y_P (e.g., $[\times, 2, \div, \times, 10, -, 10, 1, 2]$). Each symbol y in Y_P is taken from a vocabulary V_P composed of the operator set V_O (e.g., $\{+, -, \times, \div\}$), numeric constant set N_C (e.g., $1, \pi$), and numeric values N_P in X_P, i.e., $V_P = V_O \cup N_C \cup N_P$. As N_P varies with X_P, different problems may have different V_P.

IV. KNOS: Knowledge Guided Solver

In this section, we introduce our basic Knowledge-guided Solver (KNOS) framework that empowers large language models (LLMs) to simulate how humans *Invoke-Verify-Inject* knowledge to conduct reasoning. Specifically, dual process theory states that the structure of human cognition consists of two systems [24], [25]. System 1 is responsible for providing information such as knowledge quickly and heuristically, while System 2 slowly and analytically applies the knowledge to achieve logical and reliable reasoning.

Drawing this insight, as shown in Fig. 2, we construct two systems in KNOS: *Knowledge System* and *Inference System*, which simulate the functions of System 1 and System 2 respectively. Given problem P, they interact continuously to reason expression Y_P. To be specific, at each reasoning step t ($t = 1, 2, \ldots, m$), the Knowledge System *invokes* the necessary knowledge K_t for single-step inference based on the expression already generated by the Inference System. Then, the Inference System *verifies* K_t and *injects* it to deduce symbol y_t. The derived y_t is then sent back to Knowledge System to obtain knowledge K_{t+1} to support the next step $t + 1$. We will detail these two systems below.

A. Knowledge System

The main function of Knowledge System is to be a knowledge provider [59], so as shown in Fig. 2, it contains a knowledge base and a knowledge invoker. Since LLMs have been verified to remember a lot of knowledge from their training data in the parameters [8], here we adopt a frozen LLM as the knowledge base. For generality, it can be implemented with GPT3, ChatGPT, GPT4, etc.

Given the knowledge base, at each step t, the knowledge invoker elicits the knowledge K_t needed for single-step reasoning according to the problem sentence X_P and the generated partial expression $Y_P^t \triangleq [y_1, y_2, \ldots, y_{t-1}]$. Compared with existing works [9], [44], [45], [46] that allow LLMs to express knowledge freely throughout the entire reasoning process, we elicit knowledge from LLMs in a single step with strict mathematical significance, i.e., one symbol in Y_P corresponds to one reasoning step. Therefore, our method can enhance the pertinence of the knowledge and thus improves the accuracy of reasoning more efficiently, as well as enabling better interpretability.

Specifically, since the knowledge of the LLM is implicitly encoded in its parameters, the invoker elicits knowledge indirectly by having it reason the current step t. We design a text prompt that inputs the problem X_P and the expression Y_P^t and asks for completing one step for the target prefix expression Y_P:

$$K_t \leftarrow LLM(Prompt(X_P;\ Y_P^t)). \tag{1}$$

Prompt: Given problem "X_P", tell me how to complete a Prefix expression and explain the reason: 'answer = [Y_P^t,

Fig. 3. Prompt in (1) for GPTs at reasoning step t in KNOS' knowledge invoker.

The textual response from the LLM is denoted as knowledge K_t that will be provided for the Inference System (e.g., "To complete the... of teams." in Fig. 2(a)). For illustration, we show our prompt for GPTs (GPT3, ChatGPT, GPT4) in Fig. 3. To promote explainability, we also let the LLM give reasons to explain its output. We present two detailed examples that contain K_t and the corresponding response for each step $t \in \{1, \ldots, m\}$ in Appendix 1, available online.

B. Inference System

Inference System is responsible for applying the knowledge from Knowledge System to reason the expression Y_P. Here we aim to design a general framework that can benefit existing solvers, rather than proposing a specific model structure. Therefore, as depicted in Fig. 2, we first unify existing solvers into a "backbone solver". Then we contribute a knowledge verifier and a knowledge injector. Backbone solver encodes the problem X_P and controls the overall reasoning process for Y_P along a step-by-step symbol generation manner. At each step t, our knowledge verifier evaluates the rationality of the knowledge K_t generated by the Knowledge System and keeps the relevant ones. Then, the knowledge injector integrates the retained knowledge to backbone solver to derive y_t. In the following, we will describe these three components in turn.

1) Backbone Solver: Given problem P, the backbone solver first encodes the problem sentence X_P into word representations $\mathbf{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n]$ and generates an initial reasoning state $\boldsymbol{s}_1$:

$$(\mathbf{H} \in \mathbb{R}^{n \times d}, \boldsymbol{s}_1 \in \mathbb{R}^d) = \mathrm{Enc}([\boldsymbol{x}_i, i = 1, 2, \ldots, n]). \quad (2)$$

$\mathrm{Enc}(\cdot)$ can be implemented with different networks, ranging from RNN, pre-trained language models, MWP encoder, to LLMs [29], [32], [38], [60]. Then, the backbone solver derives expression Y_P step by step. At step t, it first predicts a symbol y_t based on the current reasoning state $\boldsymbol{s}_t$ and then generates the next reasoning state $\boldsymbol{s}_{t+1}$:

$$\begin{aligned} P(y_t) &= \mathrm{softmax}(\mathrm{Dec}_1(y_1, \ldots, y_{t-1}, \boldsymbol{s}_t, \mathrm{options})), \\ \boldsymbol{s}_{t+1} &= \mathrm{Dec}_2(\boldsymbol{s}_t, \boldsymbol{e}(y_t), \mathrm{options}). \end{aligned} \quad (3)$$

Networks $\mathrm{Dec}_1, \mathrm{Dec}_2$ vary from different MWP solvers (e.g., output gate and forget gate of LSTM respectively in DNS [2]). $\boldsymbol{e}(y_t)$ is the representation of y_t, which equals $\boldsymbol{h}_i$ if y_t is a number in N_P or an additional embedding $\boldsymbol{o}_c$ if y_t is operator c. options are optional terms in some solvers (e.g., context vector in GTS [32]). Based on (3), the symbol prediction loss of backbone solver is:

$$Loss_p = \sum_P \sum_t -log\ P(y_t | y_1, \ldots, y_{t-1}, P). \quad (4)$$

2) Knowledge Verifier: Now we discuss how KNOS evaluates the quality of knowledge K_t invoked by Knowledge System at each step t. In the literature [22], [44], previous studies have found that humans can naturally judge whether the knowledge is valuable by referring to the current reasoning state $\boldsymbol{s}_t$. For example, in Fig. 2, if the first state $\boldsymbol{s}_1$ represents "multiply the number of unique pairs of teams by the number of games played between each pair.", we can easily determine that relational knowledge about "twice"-"×", "twice"-"2" are relevant, while "football"-"team" is true but irrelevant and thus should be filtered to avoid redundancy.

Therefore, at step t, since K_t generated by LLM is represented as a natural language text (1), which can be denoted as $K_t = [k_t^1, k_t^2, \ldots, k_t^r]$ (length r varies at different step t), we first encode it into an overall hidden representation $\boldsymbol{k}_t \in \mathbb{R}^d$ by:

$$\begin{aligned} \mathbf{K}_t &\triangleq [\boldsymbol{k}_t^1, \boldsymbol{k}_t^2, \ldots, \boldsymbol{k}_t^r] = f_\theta([k_t^i, i = 1, 2, \ldots, r]), \\ \boldsymbol{k}_t &= \frac{1}{2}(\boldsymbol{k}_t^1 + \boldsymbol{k}_t^r). \end{aligned} \quad (5)$$

f_θ is a textual encoder that can be implemented with BERT [60], T5 [61], and so on. Then, we view the verification as a dichotomous task and utilize $\boldsymbol{s}_t$ to compute the probability that K_t contains the information needed for reasoning in this step:

$$\begin{aligned} \mathrm{score} &= tanh(\boldsymbol{W}_{v1} \cdot [\boldsymbol{s}_t, \boldsymbol{k}_t] + \boldsymbol{b}_{v1}), \\ p_t &= \sigma(\boldsymbol{W}_{v2} \cdot \mathrm{score} + \boldsymbol{b}_{v2}), \end{aligned} \quad (6)$$

where $\boldsymbol{W}_\cdot, \boldsymbol{b}_\cdot$ are parameters, $[\cdot]$ represents the concatenation of vectors, and $\sigma(\cdot)$ is the sigmoid function. On this basis, if p_t is larger than a threshold $\varepsilon \in [0, 1]$, the invoked knowledge K_t is retained, otherwise it will be filtered. To optimize the parameters in (5) and (6), the overall training objective of KNOS is:

$$\begin{aligned} Loss_{KNOS} &= Loss_p + \alpha_1 \cdot Loss_v, \\ Loss_v &= -\sum_P \sum_t \delta_t \cdot log\ p_t. \end{aligned} \quad (7)$$

Hyper-parameter α_1 balances symbol prediction loss $Loss_p$ and verification loss $Loss_v$. $\delta_t \in \{0, 1\}$ is the ground truth label of whether K_t is valid (details will be explained in Section VI-A2).

3) Knowledge Injector: Finally, we introduce how to inject the retained knowledge K_t into the backbone solver to guide the symbol prediction. Specifically, we draw insight from the dual mechanism of human cognition [26]. On one hand, the essential knowledge in the textual expression K_t is often sparse. Humans tend to focus on the most important parts of it according to the reasoning state. On the other hand, based on the knowledge, humans are able to find the most relevant information in the problem sentence (e.g., turn to find the number of "teams"), which can further reduce the difficulty of reasoning. Therefore, at step t, we capture the dual attention of reasoning state on knowledge (denoted as k_i^{attn}) and knowledge on problem sentence (denoted as h_i^{attn}):

$$\begin{aligned} k_i^{attn} &= \mathrm{softmax}(\boldsymbol{W}_{k1} \cdot tanh(\boldsymbol{W}_{k2} \cdot [\boldsymbol{s}_t, \boldsymbol{k}_t^i])), \\ h_i^{attn} &= \mathrm{softmax}(\boldsymbol{W}_{h1} \cdot tanh(\boldsymbol{W}_{h2} \cdot [\boldsymbol{k}_t, \boldsymbol{h}_i])), \end{aligned} \quad (8)$$

Then we integrate these two kinds of information into the current reasoning state $\boldsymbol{s}_t$ by:

$$\boldsymbol{s}'_t = \boldsymbol{s}_t + \beta_1 \cdot \Sigma_{i=1}^{r} k_i^{attn} \cdot \boldsymbol{k}_t^i + \beta_2 \cdot \Sigma_{i=1}^{n} h_i^{attn} \cdot \boldsymbol{h}_i. \quad (9)$$

β_1, β_2 are hyper-parameters that balance the attention weights. The updated $\boldsymbol{s}'_t$ that has fused the knowledge will be ultimately used in (3) to predict symbol y_t and generate next reasoning state $\boldsymbol{s}_{t+1}$.

V. EKNOS: External Knowledge Enhancement

In KNOS, we have achieved a basic framework that utilizes LLMs to simulate human cognitive processes including knowledge invocation, verification, and injection. However, a LLM's mathematical knowledge is entirely acquired from its training data, just like a student taking a *closed-book* exam, which may encounter deficiencies in practice (e.g., lack of math formulas for MWP). This may limit the effectiveness of KNOS as LLMs serve as the knowledge base in Knowledge System that guides the overall reasoning. Thus, in this section, we extend KNOS to EKNOS by considering an *open-book* scenario, where we specifically introduce two kinds of external knowledge to enhance its reasoning ability.

Commonsense Knowledge: As shown in the red line in Fig. 2, we first consider the utilization of commonsense knowledge, which can complement both the understanding of the problem and the derivation of symbols [62], [63], [64]. For instance, the knowledge between "twice" and "×" is vital to understand the meaning of calculating the "total games" and help derive "×". For this purpose, we construct an external knowledge graph $KG = (V, E)$ in Knowledge System. Its vertices V consist of words and operators (e.g., $\{+, -, \times, \div\}$) in MWP. Its edges E are composed of two kinds of relations, one is the relationship among words (denoted as $ww_{i,j}$ between words i and j), and the other is the relationship between words and operators ($wo_{i,c}$ between word i and operator c). Word-word relationship comes from HowNet [65] and ConceptNet [66]. Word-operator relationship comes from the graph published by [67]. To further ensure rationality and reduce redundancy, we combine these graphs by only keeping their shared words in V and "RelatedTo" relation of ConceptNet as $ww_{i,j}$ in E. Given problem P, we select a subgraph $KG_P = (V_P, E_P)$ from KG, which takes the words in P and all operators in V as nodes V_P and their edges in E as E_P.

Math Formula Knowledge: In addition to commonsense, math formulas constitute a fundamental aspect of subject knowledge, serving as the core for guiding mathematical thinking and logic in human reasoning [68], [69], [70]. Therefore, we introduce a set of formulas (denoted as F) published by [22] as a domain knowledge base, which is specifically built for MWP. Each formula $f \in F$ is expressed in natural language (e.g., "$C_n^2 = n \times (n-1) \div 2$").

On this basis, considering that reasoning on different steps may require different formulas, we design a selector to extract the most suitable formula $f_t \in F$ that should be used at each step $t \in \{1, 2, \ldots, m\}$ from F. To be specific, at step t, we calculate the probability of choosing formula $f \in F$ using the reasoning state $\boldsymbol{s}_t$ and a representation $\boldsymbol{r}_f \in \mathbb{R}^d$ of f as follows:

$$\begin{aligned} att_f(i) &= \text{softmax}(\boldsymbol{W}_{a1} \cdot tanh(\boldsymbol{W}_{a2} \cdot [\boldsymbol{r}_f, \boldsymbol{h}_i])), \\ att_s^t(i) &= \text{softmax}(\boldsymbol{W}_{b1} \cdot tanh(\boldsymbol{W}_{b2} \cdot [\boldsymbol{s}_t, \boldsymbol{h}_i])), \\ \boldsymbol{c}_f^t &= \sum_{i=1}^{n} (att_f(i) + \mu \cdot att_s^t(i)) \cdot \boldsymbol{h}_i, \\ p_{ret}^t(f) &= \text{softmax}(\boldsymbol{W}_{s1} \cdot tanh(\boldsymbol{W}_{s2} \cdot \boldsymbol{c}_f^t)), \end{aligned} \quad (10)$$

where $\boldsymbol{W}_{a\cdot}, \boldsymbol{W}_{b\cdot}, \boldsymbol{W}_{s\cdot}$ are newly introduced parameters, $\boldsymbol{r}_f$ is pretrained by [22], μ is a hyper-parameter controlling the preference of attention. Then, we adopt f with the highest probability $p_{ret}^t(f)$ as f_t. To train parameters in (10), we introduce a formula selection loss $Loss_r$, thus the training loss of EKNOS is:

$$\begin{aligned} Loss_{EKNOS} &= Loss_{KNOS} + \alpha_2 \cdot Loss_f, \\ Loss_f &= -\sum_P \sum_t \sum_f \zeta_f^t \cdot log\, p_{ret}^t(f). \end{aligned} \quad (11)$$

α_2 is another weighting hyper-parameter. $\zeta_f^t \in \{0, 1\}$ is the label indicating whether step t should select the formula f.

Without loss of generality, in (10), we select the (only one) formula with the highest probability $p_{ret}^t(f)$. To adapt to scenarios where multiple distinct formulas are needed, we could modify the softmax function to independently evaluate the probability of each formula being selected, enabling our framework to handle steps requiring multiple formulas without altering the overall flow.

Knowledge Enhancement: At step t, after obtaining the commonsense knowledge KG_P and formula f_t, we convert them into a textual input for knowledge invoker to elicit more comprehensive knowledge from the LLM to enhance guiding the Inference System. Specifically, in order to improve the extensibility of our framework, for KG_P, we focus on the existence of its edges without refining their specific semantics (e.g., hypernymy [71], [72]). Therefore, we convert the edge between any pair of nodes A, B to the text "A-B". For example, for knowledge between "twice" and "×", we convert to the text "twice-×". For formula f_t, we leave it unchanged because it has been stored as a text. On this basis, we ask the LLM in knowledge invoker to apply these knowledge by prompts:

$$K_t \leftarrow LLM(Prompt(X_P;\ Y_P^t;\ KG_P;\ f_t)). \quad (12)$$

Fig. 4 shows the detailed prompt, whose response will ultimately be used as knowledge K_t for Inference System to reason y_t (please see Appendix 1 for examples), available online.

Prompt: Given problem "X_P", knowledge among words: $\{ww_{i,j} \in KG_P\}$, knowledge between words and operators: $\{wo_{i,j} \in KG_P\}$, knowledge of formulas: f_t, tell me how to complete a Prefix expression and explain how you apply the knowledge to get the result: 'answer = [Y_P^t,

Fig. 4. Prompt in (12) for GPTs at reasoning step t in EKNOS' knowledge invoker. (Red: the knowledge enhancement compared with KNOS' in Fig. 3).

TABLE II
STATISTICS OF DATASETS

Dataset	Math23K	MAWPS
Num. problems	23,162	2,373
Num. operators	5	4
Avg. problem length	28.02	30.08
Avg. expr. length	5.55	4.20
Num. nodes in KG	1,806	712
Num. edges in KG	33,179	4,490
Num. formulas in F	131	46
Num. problems requiring formula	7,750	911

VI. EXPERIMENTS

To answer *Q1* in Section I, we compare the effects of EKNOS and KNOS in Sections VI-B1, VI-C2, and VI-C5. To answer *Q2*, we implement representative LLMs in our frameworks and explore their reasoning accuracy, efficiency, and interpretability in Sections VI-B, VI-C1, VI-C3, VI-C4, and VI-C6.[2]

A. Experimental Setup

1) Dataset Description: Since our core goal is to use LLMs to simulate the process of human *Invoke-Verify-Inject* knowledge at strict mathematical step level, we conduct experiments on two datasets where the mathematical steps are clear. Math23K [2] is a widely studied Chinese benchmark that contains 23,162 problems. MAWPS [19] is an English dataset that contains problems with one or more unknown variables. We select 2,373 problems with only one variable. Table II summarizes their basic statistics. Besides, since different datasets contain different words and operators, their corresponding external knowledge bases (i.e., KG, F) in EKNOS are also different. Thus, based on [22], [65], [66], [67], we report their statistics in Table II, from which we can see that our considered knowledge are very common and will be verified important for answer reasoning in Sections VI-B and VI-C.

2) Implementation Details: Our KNOS and EKNOS are general frameworks that can be applied to different LLMs. To fully verify their effectiveness, in Knowledge System, we implement the knowledge base with GPT3 (text-davinci-002), ChatGPT (gpt-3.5-turbo-0125), and GPT4 (gpt-4-0125-preview). In Inference System, we implement the Backbone Solver as the most basic model GTS [32] in MWP research. Encoder f_θ in (5) is set as BERT for Math23K and T5 for MAWPS. The dimension d of word/knowledge representation and reasoning state is set to 768 and 1,024 for Math23K and MAWPS, respectively. For hyper-parameters, in (7) and (11), α_1, α_2 are 0.1, 0.05. In (9), both β_1, β_2 are set to 0.5. In (10), μ is set as 0.02 according to [22]. To train the knowledge verifier in Section IV-B2, we ask GPT4 to determine whether the meaning of K_t is consistent with the next symbol in expression Y_P, setting the ground truth label δ_t in (7) to 1 if so, and 0 otherwise. The threshold ε is set as 0.5. To train the formula selector, we use the labels annotated in [22] as ζ_f^t in (11). For dataset preprocessing, we follow the public train/valid/test partition for Math23K. For MAWPS, the models are evaluated with 5-fold cross-validation. All experiments are run on a Linux server with four 2.30 GHz Intel Xeon Gold 5218 CPUs and a Tesla A40 GPU.

3) Baselines: We choose 11 representative methods as baselines, including state-of-the-art MWP models, GPT3, ChatGPT, and GPT4:

- *DNS* [2]: uses a vanilla seq2seq model to translate math problems into equations directly.
- *GTS* [32]: adopts a top-down goal decomposition manner to generate binary expression trees.
- *Graph2Tree* [29]: strengthens problem understanding with relationships and order among quantities.
- *KA-S2T* [73]: uses external commonsense knowledge graphs to enrich problem comprehension.
- *BERT-Tree* [74]: utilizes BERT as problem encoder and derives expressions by the decoder of GTS.
- *MWP-BERT* [33]: designs numeracy grounded pre-training objectives for language models.
- *LogicSolver* [37]: retrieves math formulas as prompts for problem encoding and post-explanations.
- *FOMAS* [22]: elaborates three mechanisms that apply math formulas to guide symbol prediction.
- *GPT3* [6]: is a basic LLM with 175 billion parameters trained on high-quality web crawl data.
- *ChatGPT*: is one of the most popular LLMs trained with dialogue data and RLHF technique. We test its performance in June, 2024.
- *GPT4* [7]: is the current SOTA LLM with human-level performance on many difficult benchmarks. We test its performance in June, 2024.

To more directly validate our methods, we also introduce 6 current LLM-enhanced methods. All LLM-based baselines are combined with Self-Consistency [47] to ensure the stability. Due to the cost of calling the GPT4 API, we only apply these methods to GPT3 and ChatGPT:

- *Standard Prompting (SP)* [8]: employs an one-shot "(Problem, Answer)" example in the prompt.
- *Chain-of-Thought (CoT)* [9]: asks to generate intermediate reasoning process step by step.
- *Tree-of-Thought (ToT)* [44]: performs tree-structured reasoning that considers multiple different reasoning paths and self-evaluation.
- *Program-of-Thought (PoT)* [75]: generates text and programming statements in the solution.
- *Program-Aided Language models (PAL)* [76]: generates Python programs as the intermediate steps and uses code interpreter to derive the answer.
- *Knowledge RAG (KnowRAG)* [51]: Following the fundamental paradigm of RAG, this baseline provides all the selected knowledge in EKNOS to LLMs. The prompt is "Given knowledge: $KG_P, \{f_t, t = 1, \ldots, m\}$, solve the following problem: X_P".

B. Performance on Answer Reasoning

1) Answer Accuracy: Table III shows the answer accuracy of all methods, and we find several key observations. First, we can

[2]Our codes are available at https://github.com/Ljyustc/KNOS.

TABLE III
ANSWER ACCURACY ($*/**: P < 0.05/0.001$ W.R.T. SOTA MWP BASELINE; $\dagger/\ddagger: P < 0.05/0.001$ W.R.T. SOTA LLM)

Method	Type	External Knowledge resources	Math23K	MAWPS
DNS	MWP model	-	0.581	0.595
GTS	MWP model	-	0.756	0.826
Graph2Tree	MWP model	-	0.774	0.837
KA-S2T	MWP model	Commonsense knowledge graph	0.763	/
BERT-Tree	MWP model	-	0.833	0.872
MWP-BERT	MWP model	-	0.836	0.829
LogicSolver	MWP model	Math Formulas	0.834	/
FOMAS	MWP model	Math Formulas	0.848	0.886
GPT3	LLM	-	0.181	0.643
GPT3(SP)	LLM	-	0.250	0.662
GPT3(CoT)	LLM	-	0.311	0.749
GPT3(KnowRAG)	LLM	Commonsense knowledge graph + Math formulas	0.280	0.673
ChatGPT	LLM	-	0.659	0.906
ChatGPT(SP)	LLM	-	0.585	0.897
ChatGPT(CoT)	LLM	-	0.669	0.903
ChatGPT(ToT)	LLM	-	0.595	0.918
ChatGPT(PoT)	LLM	-	0.501	0.701
ChatGPT(PAL)	LLM	-	0.717	0.919
ChatGPT(KnowRAG)	LLM	Commonsense knowledge graph + Math formulas	0.670	0.911
GPT4	LLM	-	0.729	0.941
GPT4(KnowRAG)	LLM	Commonsense knowledge graph + Math formulas	0.756	0.943
KNOS(GPT3)	LLM	-	$\mathbf{0.838}^{\ddagger}$	$\mathbf{0.894}^{*\ddagger}$
EKNOS(GPT3)	LLM	Commonsense knowledge graph + Math formulas	$\mathbf{0.853}^{*\ddagger}$	$\mathbf{0.901}^{**\ddagger}$
KNOS(ChatGPT)	LLM	-	$\mathbf{0.845}^{\ddagger}$	$\mathbf{0.915}^{**}$
EKNOS(ChatGPT)	LLM	Commonsense knowledge graph + Math formulas	$\mathbf{0.858}^{*\ddagger}$	$\mathbf{0.921}^{**\dagger}$
KNOS(GPT4)	LLM	-	$\mathbf{0.863}^{**\ddagger}$	$\mathbf{0.954}^{**\dagger}$
EKNOS(GPT4)	LLM	Commonsense knowledge graph + Math formulas	$\mathbf{0.874}^{**\ddagger}$	$\mathbf{0.957}^{**\ddagger}$

see that our KNOS and EKNOS outperform all the baselines. By applying paired t-test, the improvements over the SOTA MWP model FOMAS and LLMs are statistically significant with $p < 0.001 \sim 0.05$, even when using the advanced GPT4 as the backbone. These results demonstrate the necessity and effectiveness of utilizing LLMs to simulate the process of human knowledge application for reliable mathematical reasoning. Second, whether implemented with GPT3, ChatGPT, or GPT4, our frameworks perform better than LLM-enhanced methods, especially KnowRAG that inputs knowledge as an overall prompt to LLMs. This not only confirms the generality of our frameworks, but also directly illustrates that empowering LLMs with step-by-step application of knowledge is more in line with human cognitive process and can better elicit LLMs' reasoning abilities. Third, combining the observation that EKNOS improves the accuracy of KNOS and that KnowRAG surpasses the original LLMs, we conclude that it is necessary to introduce external knowledge into LLMs, underscoring their inherent limitations in fully mastering specialized knowledge. Besides, it also validates the efficacy of our knowledge enhancement.

2) Ablation Study: In this section, we perform ablation study and report the result of each case in Table IV. Specifically, in Knowledge System, since knowledge invoker is a necessary component, we investigate the external knowledge enhancement of EKNOS and omit the commonsense knowledge graph (denoted as "w/o KG") and formula knowledge (denoted as "w/o Formula"). In Inference System, since knowledge injection is also a necessary process, we eliminate only the knowledge verifier (denoted as "w/o Verify").

TABLE IV
ABLATION STUDY

	Math23K		
EKNOS	GPT3	ChatGPT	GPT4
	0.853	**0.858**	**0.874**
w/o KG	0.845	0.854	0.866
w/o Formula	0.840	0.851	0.864
w/o Verify	0.836	0.847	0.852
	MAWPS		
EKNOS	GPT3	ChatGPT	GPT4
	0.901	**0.921**	**0.957**
w/o KG	0.898	0.911	0.951
w/o Formula	0.890	0.911	0.944
w/o Verify	0.887	0.906	0.930

First, all components of EKNOS contribute to the knowledge application of LLMs in carrying out mathematical reasoning, since removing each of them results in performance degradation. Second, the removal of knowledge verifier has the greatest impact on the effect, meaning that ensuring the quality of knowledge is the basis of correctly applying it by LLMs, while redundant or faulty knowledge may cause the negative effect.

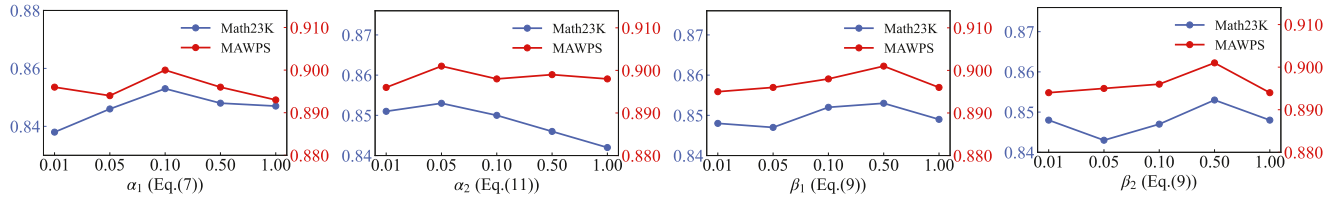

Fig. 5. Answer Accuracy with different $\alpha_1, \alpha_2, \beta_1, \beta_2$.

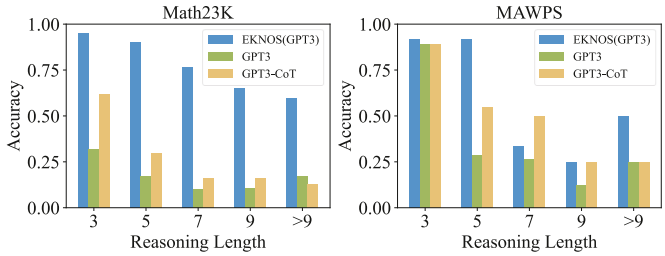

Fig. 6. Answer Accuracy over reasoning length.

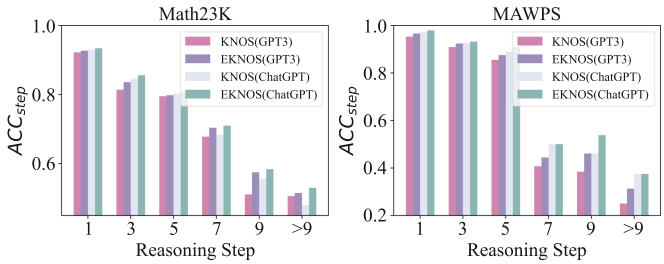

Fig. 7. Single-step accuracy ACC_{step}.

Third, "w/o Formula" diminishes the effect more greatly than "w/o KG", which suggests that the math formula contributes more to solving MWP. At the same time, this reveals that there is still room for improvement in the learning of domain knowledge by the current LLMs.

3) Hyper-Parameter Sensitivity: In our frameworks, α_1, α_2 in (7) (11) and β_1, β_2 in (9) play an important role for modeling. Specifically, α_1, α_2 balance the training weight of symbol prediction, knowledge verification, and formula selection. β_1, β_2 control the preference to focus more on the invoked knowledge or the problem itself in knowledge injection. Therefore, we show the performances of $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \{0.01, 0.05, 0.1, 0.5, 1\}$ in Fig. 5, where we take the basic GPT3 as the LLM in our EKNOS.

First, both the answer accuracies of α_1, α_2 show a trend of first rising and then falling. This suggests that it is necessary to balance the objectives of knowledge verification and injection. Besides, the peak of performance is attained when $\alpha_1 = 0.1$ and $\alpha_2 = 0.05$, which again demonstrates that knowledge verification plays a more important role in our frameworks. Second, β_1, β_2 show a similar trend, also illustrating the need to strike a precise balance between knowledge and problem text when injecting knowledge. In particular, they both perform best when equal to 0.5, indicating that the two are equally important. This observation validates the necessity and validity of modeling dual mechanism of human cognition in our knowledge injector.

C. Framework Analysis

1) Performance Over Reasoning Length: Furthermore, we investigate the performance of our frameworks as the reasoning length increases. Since we treat a single symbol derivation as a single reasoning step, the reasoning length is equivalent to the length of the target expression. Therefore, we report the accuracy of our EKNOS(GPT3), GPT3 and GPT3-CoT on test samples with expressions of different lengths in Fig. 6.

We first observe that generally, the performance of each method degrades with the increasing reasoning length, which is reasonable due to increasing difficulty. Second, our EKNOS(GPT3) improves the performances of GPT3 and GPT3-CoT over almost all reasoning lengths, which validates the robustness of our framework under varying problem complexity. Third, for problems with reasoning length of 5, our framework brings the highest improvement in accuracy (73.1%/63.2% on Math23 K/MAWPS). We think this is because our framework injects the knowledge of LLMs step by step, so that the advantage is more significant in problems with a larger number of reasoning steps. For those with length greater than 5, due to the small number of these problems, the occasional factors have a great impact on the results, which may not be representative.

2) Effectiveness of Knowledge Enhancement: To deeply explore the necessity of external knowledge for stimulating LLMs' reasoning ability, in addition to comparing the overall accuracy in Section VI-B1, here we investigate from the perspective of single-step accuracy since we inject knowledge at a step-wise level. Specifically, we calculate the accuracy (denoted as ACC_{step}) of all test problems at steps 1, 3, 5, 7, 9, and steps over 9, and compare the results of our EKNOS and KNOS (with GPT3 and ChatGPT as the knowledge base) in Fig. 7. Notably, even if the model made an error in an earlier step, we still count the problem in the subsequent steps, which aligns with similar works [77], [78] that also investigate step-wise accuracy.

First, we observe that after external knowledge enhancement, ACC_{step} is obviously improved for both GPT3 and ChatGPT for all steps. This directly illustrates the necessity of supplementing the knowledge for LLMs, especially in a single mathematical step to enhance their reasoning ability. Second, the effect shows a higher boost on GPT3. This is probably because GPT3 does not have as much knowledge as ChatGPT and therefore relies

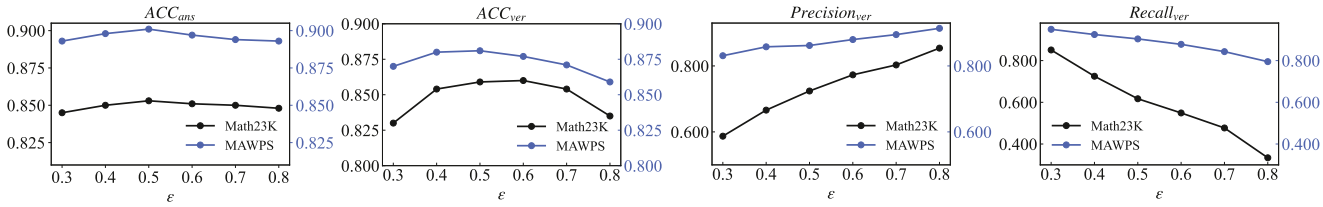

Fig. 8. Performances of threshold ε in knowledge verifier.

more on external knowledge bases. Third, in steps larger than 7, the introduction of external knowledge leads to less reduction in accuracy. This shows that our developed external knowledge enhancement can improve the stability of LLMs' reasoning ability. The above phenomena prove the necessity and validity of our proposed EKNOS framework.

3) Analysis of Knowledge Verifier: From the aforementioned experiments, knowledge verifier plays an important role in our frameworks, so we first explore the influence of its core threshold ε. We take $\varepsilon \in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$ and investigate from two perspectives: 1) answer accuracy (ACC_{ans}), 2) verification performance which we view as a dichotomous task measured by accuracy (ACC_{ver}), precision ($Precision_{ver}$), and recall ($Recall_{ver}$). Here, we also adopt GPT3 for basic exploration.

From Fig. 8, under the experimental setup $\varepsilon = 0.5$ (Section VI-A2), our EKNOS(GPT3) achieves satisfying results (e.g., all metrics are around 0.9 on MAWPS) in both answer reasoning and knowledge verification. It indicates that our framework can effectively optimize both at the same time. Second, ACC_{ans} and ACC_{ver} show an increasing trend when ε increases from 0.3 to 0.5, but decline immediately if ε continues to increase. On one hand, this verifies that there is a strong positive correlation between the performances of answer reasoning and knowledge verification, both of which are sensitive to ε. On the other hand, the results of ACC_{ans} specifically illustrate that applying excessive incorrect knowledge or missing essential knowledge is detrimental to mathematical reasoning. Third, the results of knowledge verification on MAWPS are better than those on Math23K under all ε, which may be because the current LLMs' training data is mainly in English, so it is more sensitive to problems and knowledge represented in English. Although the performance on Math23K is slightly lower (but with all metrics above 0.6), our KNOS(GPT3) still shows a 65.7% improvement over the original GPT3 in Table III, along with great gains observed for ChatGPT and GPT4. This phenomenon reflects that our frameworks are robust enough to adapt to potential verification deviations. We believe it is mainly due to that in *knowledge injector*, we draw insight from the dual mechanisms of human cognition to design a dual attention mechanism in (8). This mechanism dynamically adjusts the influence between verified knowledge and the problem. Additionally, in (9), our reasoning state s'_t is calculated based on an initial state s_t without any verified knowledge, which can minimize the impact of incorrect knowledge as much as possible.

To delve deeper into the performance of knowledge verifier, we conduct a further analysis of the rest 8.5% errors of KNOS(ChatGPT) on MAWPS and find the following

TABLE V
TWO EXAMPLES FOR ANALYZING THE KNOWLEDGE VERIFIER

Problem 1	It takes 4 apples to make 1 pie . How many apples does it take to make 504 pies ?
Target Expression	$\times$, 4, 504
K_2	To complete the prefix expression, we need to take 4 times as many apples to make 504 pies.
Problem 2	Bryan took a look at his books as well. If Bryan has 56 books in each of his 9 bookshelves , how many books does he have in total ?
Target Expression	$\times$, 56, 9
K_2	To complete the prefix expression, we need to find the number of bookshelves, which is 9.

observations. 1) For cases where ChatGPT-invoked knowledge is correct but knowledge verifier considers it incorrect, about 63% are due to that the invoked knowledge is holistic and not clearly expressed at step level. Take the problem 1 in Table V as an example. The knowledge K_2 invoked by ChatGPT at step 2 is overly broad and does not clearly express to reason the number "4". Therefore, our knowledge verifier considers the knowledge to be irrational. 2) For cases where ChatGPT-invoked knowledge is incorrect but knowledge verifier considers it correct, about 86% are due to that a reasoning step may have multiple correct outputs. Considering the problem 2, the knowledge K_2 generated by ChatGPT is about inferring the number "9", which is actually reasonable. However, the ground-truth label of whether the LLM-invoked knowledge is correct is determined by GPT4 comparing it with the annotated symbol in target expression (explained in Section VI-A2). Thus, since the annotated second symbol is "56", the ground-truth label of knowledge invoked by ChatGPT is considered "incorrect".

Based on these observations, we summarize two potential directions to explore in the future: 1) How to adjust the granularity of LLM-invoked knowledge, which can be helpful to improve the controllability of LLMs, 2) How to make LLMs master equivalent solutions for a problem, which may be a key towards authentic mathematical reasoning ability.

4) Cost Efficiency: The key advantage of our frameworks is using large models to simulate the human knowledge application at a strict mathematical step level, thereby ensuring the relevance and high utility of knowledge. However, it may raise concerns about the need to access the LLM API multiple times and the real runtime/token consumption when solving a single problem. To address this issue, we compare our frameworks with Tree-of-Thought (ToT), which also requires several API calls to the LLMs. Due to potential instability in API access rates, tracking accurate runtime can be challenging and may lead to

TABLE VI
PROMPT OF TREE-OF-THOUGHT (TOT)

Initial Prompt	Please answer the following math word problem: question: "X_P"\n Make a Plan then answer. Your output should be of the following format:\n Plan: \n Your plan here.\n Answer:\n Your answer here
Score Prompt	Analyze the following answer, then at the last line conclude 'Thus the score is s', where s is an integer from 1 to 10.\n {Answer}
Vote Prompt	Given an instruction and several choices, decide which choice is most promising. Analyze each choice in detail then conclude in the last line 'The best choice is s', where s the integer id of the choice. \n Instruction: {Instruction}; Choices:{Choices}

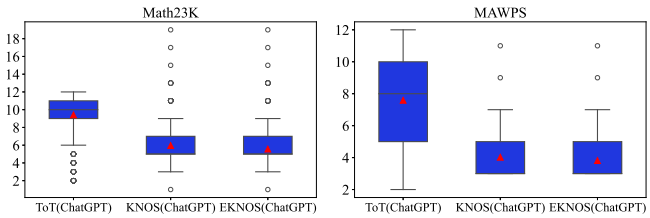

Fig. 9. The number of API calls for different methods.

unfair comparisons between methods. Therefore, we investigate from two perspectives: 1) *The length of prompt*, which reflects the extent of reliance on manual prompt engineering, and 2) *The number of API calls required to solve a single problem*, which directly impacts the cost of practical application. The combination of these indicators can sufficiently reflect real-time performance including runtime speed and cost, while considering them separately allows for a deeper analysis of each method's performance.

First, the prompt of ToT is composed of three parts: "Initial Prompt", "Score Prompt", and "Vote Prompt", which as shown in Table VI, contains 91 prompting words. In comparison, our prompts for KNOS and EKNOS only contain 15 and 33 words as shown in Figs. 3 and 4 respectively, which require nearly half the prompt tokens and are much easier to design and use. Second, we visualize the average number of API calls required by different methods in Fig. 9. It is evident that our frameworks require significantly fewer calls (i.e., $5.6 \sim 5.8$ on Math23K, $3.8 \sim 3.9$ on MAWPS) compared to ToT (9.1 and 7.5 respectively). These findings suggest that, under the same conditions, our frameworks can reduce the costs by half. Combined with the comparison of the accuracy performances in Table III, our frameworks can ensure accuracy, efficiency, and scalability, simultaneously.

5) Generalization Ability: In previous experiments, we used Math23 K and MAWPS datasets with explicit mathematical steps (i.e., expressions) to validate the necessity and effectiveness of our frameworks in enabling LLMs to utilize knowledge step by step. To demonstrate that our frameworks are not limited to these types of data, we test them without any additional training or fine-tuning on a more challenging MWP dataset, GSM8K [79], which does not use expressions as output, as the test set to verify the generalization ability of our methods. For this validation, we use the SOTA MWP model BERT-Tree [74], ChatGPT series models, and an advanced open-source LLM Deepseek-V3 [80] as baselines.

TABLE VII
GENERALIZATION PERFORMANCE ON GSM8K DATASET

	Answer Accuracy
BERT-Tree[1]	0.138
ChatGPT	0.813
ChatGPT(SP)	0.810
ChatGPT(CoT)	0.820
ChatGPT(ToT)	0.834
ChatGPT(PoT)	0.785
ChatGPT(PAL)	0.813
Deepseek-V3	0.965
KNOS(ChatGPT)	**0.846**
EKNOS(ChatGPT)	**0.853**
KNOS(Deepseek-V3)	**0.983**
EKNOS(Deepseek-V3)	**0.988**

[1]Since BERT-Tree needs expressions for training, we used GPT4 to convert GSM8K into "Problem-Expression" format for its experiment.

From Table VII, we can see that our KNOS enhances the reasoning capabilities of ChatGPT/Deepseek for 3.3%/1.8% on this more challenging dataset. This demonstrates the generalizability of simulating human knowledge application in dual-process reasoning. In contrast, current small models struggle to solve the problems in this difficult dataset (answer accuracy of BERT-Tree is only 0.138). Furthermore, by incorporating external knowledge, our EKNOS still improves the LLM's performance for 0.5–0.7%. This indicates that large models may still lack some of the knowledge required to solve more difficult mathematical problems. Additionally, it validates the necessity and superiority of introducing open-book scenario and providing external knowledge to LLMs step by step.

6) Interpretability Verification: Finally, we conduct case study in Fig. 10 to illustrate the interpretability of our frameworks. For each case, we first report the problem sentence and the solution/expression generated by GPT3, our KNOS(GPT3) and EKNOS(GPT3). Then, we visualize the invoked knowledge K_t and verification probability p_t of our frameworks at steps where they essentially correct the mistakes of GPT3 (i.e., $t = 1$ for cases 1 and 3, $t = 3$ for case 2). Especially, we also show the external knowledge (i.e., KG_P, f_t) for EKNOS(GPT3).

For cases 1 and 2, GPT3 omits division and addition operations in its continual natural language representation of reasoning. In contrast, our frameworks gradually invoke GPT3's knowledge about "total_amount" and "scarves, gloves, clothing", and thus correctly derive "÷" and "+". This phenomenon shows the necessity and superiority of our work to elicit and apply the knowledge of LLMs at the step level of mathematical sense (i.e., symbol-by-symbol). For case 3, GPT3 and our KNOS(GPT3) both miss a division operation. The reason is that they grasp the formula of the height of the cylinder incorrectly. Comparatively, our extended EKNOS selects a specific formula "$height = area\_cylinder \div perimeter\_base$" in the first step and correctly deduces all subsequent symbols under its guidance. This shows the importance of introducing external knowledge, especially domain formulas, to improve the reasoning ability of LLMs. In particular, although KNOS invokes false knowledge

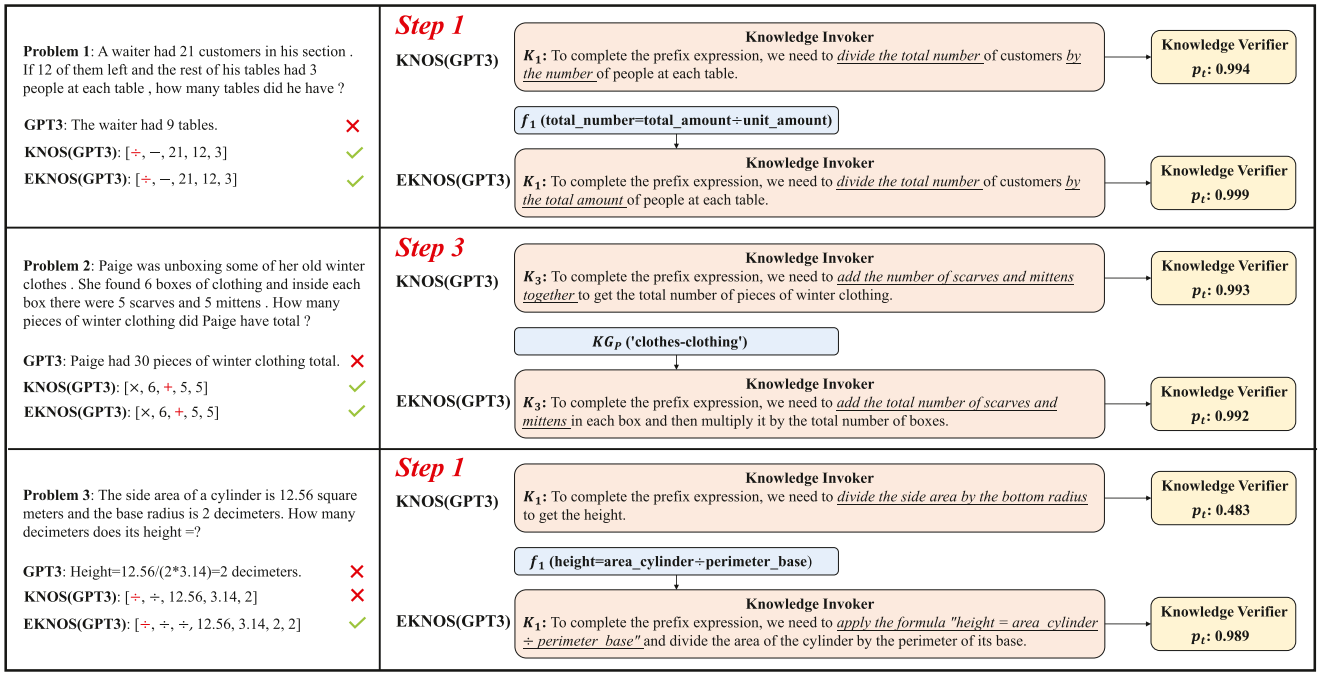

Fig. 10. Case study. We visualize the invoked knowledge K_t and verification probability p_t of our KNOS/EKNOS implemented with GPT3 at $t = 1$ for cases 1 and 3, and $t = 3$ for case 2. We also show the selected external knowledge f_t, KG_P for EKNOS.

at $t = 1$ for case 3, our knowledge verifier can still identify it correctly ($p_1 = 0.483$), while for correct knowledge in other cases it gives a passing probability of more than 0.98. These results demonstrate the effectiveness of our knowledge verifier.

VII. Conclusion and Future Work

In this paper, we studied how to enhance the mathematical reasoning capacity of LLMs by leveraging them to simulate human reasoning process for math word problem. Specifically, we proposed a novel Knowledge-guided Solver (KNOS) framework that contained Knowledge-Inference Systems inspired by human cognitive structure and conducted human-like mathematical reasoning through knowledge *Invoke-Verify-Inject* processes. Besides, we extended KNOS to EKNOS by enhancing LLMs' reasoning ability with external commonsense and domain formulas. Experiments on GPTs verified our frameworks' improvements in reasoning accuracy, efficiency, generality, and interpretability. They also validated the necessity of injecting knowledge into LLMs at the level of strict mathematical steps.

For future study, we first consider incorporate more human reasoning behaviors into our frameworks, such as planning and reflection. Second, our frameworks are designed to emulate human cognition and are highly generalizable (witnessed in Section VI-C5). We plan to expand them to more advanced LLMs (e.g., OpenAI o1) and more tasks (e.g., geometric reasoning [81]) to explore LLMs' grasp of more complex domain knowledge [82], [83], [84]. In this process, although additional training or fine-tuning may be required, we believe this extra step is necessary, because many studies have revealed that relying solely on prompt engineering may be constrained by LLMs' built-in capabilities [85], [86]. For example, ToT is more effective for tasks with specific reasoning structure [87]. In contrast, to truly build a model with strong capabilities, it is essential to incorporate corresponding modules to mimic human cognitive processes [88], [89]. We hope our work could boost more studies.

Acknowledgment

This paper was produced by the IEEE Publication Technology Group. They are in Piscataway, NJ.

References

[1] D. Zhang, L. Wang, L. Zhang, B. T. Dai, and H. T. Shen, "The gap of semantic parsing: A survey on automatic math word problem solvers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 9, pp. 2287–2305, Sep. 2020.
[2] Y. Wang, X. Liu, and S. Shi, "Deep neural solver for math word problems," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2017, pp. 845–854.
[3] J. Liu, Z. Huang, C. Zhai, and Q. Liu, "Learning by applying: A general framework for mathematical reasoning via enhancing explicit knowledge learning," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 4497–4506.
[4] P. Lu et al., "Inter-GPS: Interpretable geometry problem solving with formal language and symbolic reasoning," in *Proc. Joint Conf. Annu. Meeting Assoc. Comput. Linguistics, Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 6774–6786.
[5] T. H. Trinh et al., "Solving olympiad geometry without human demonstrations," *Nature*, vol. 625, no. 7995, pp. 476–482, 2024.
[6] T. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.
[7] J. Achiam et al., "GPT-4 technical report," 2023, *arXiv:2303.08774*.
[8] W. X. Zhao et al., "A survey of large language models," 2023, *arXiv:2303.18223*.
[9] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 24824–24837.

[10] L. Pan, A. Albalak, X. Wang, and W. Y. Wang, "Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2023, pp. 3806–3824.

[11] O. Golovneva et al., "ROSCOE: A suite of metrics for scoring step-by-step reasoning," in *Proc. 11th Int. Conf. Learn. Representations*, 2023.

[12] D. N. Ribeiro et al., "STREET: A multi-task structured reasoning and explanation benchmark," in *Proc. 11th Int. Conf. Learn. Representations*, 2023.

[13] S. Xue et al., "Decompose, analyze and rethink: Solving intricate problems with human-like reasoning cycle," in *Proc. 38th Annu. Conf. Neural Inf. Process. Syst.*, 2024, pp. 357–385.

[14] J. Qian et al., "Limitations of language models in arithmetic and symbolic induction," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2023, pp. 9285–9298.

[15] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy, "Challenges and applications of large language models," 2023, *arXiv:2307.10169*.

[16] D. Hendrycks et al., "Measuring mathematical problem solving with the math dataset," in *Proc. 35th Conf. Neural Inf. Process. Syst. Datasets Benchmarks Track*, 2021.

[17] H. Liu et al., "MathBench: Evaluating the theory and application proficiency of LLMs with a hierarchical mathematics benchmark," 2024, *arXiv:2405.12209*.

[18] X. Wang et al., "SCIBENCH: Evaluating college-level scientific problem-solving abilities of large language models," in *Proc. 41st Int. Conf. Mach. Learn.*, 2024, Art. no. 2072.

[19] R. Koncel-Kedziorski et al., "MAWPS: A math word problem repository," in *Proc. Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2016, pp. 1152–1157.

[20] B. K. Hofer and P. R. Pintrich, "The development of epistemological theories: Beliefs about knowledge and knowing and their relation to learning," *Rev. Educ. Res.*, vol. 67, no. 1, pp. 88–140, 1997.

[21] R. Audi, *Epistemology: A Contemporary Introduction to the Theory of Knowledge*. Evanston, IL, USA: Routledge, 2010.

[22] J. Liu et al., "Guiding mathematical reasoning via mastering commonsense formula knowledge," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2023, pp. 1477–1488.

[23] P. P. Ray, "ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope," *Internet of Things Cyber-Physical Syst.*, vol. 3, pp. 121–154, 2023.

[24] J. S. B. Evans, "Dual-processing accounts of reasoning, judgment, and social cognition," *Annu. Rev. Psychol.*, vol. 59, pp. 255–278, 2008.

[25] D. Kahneman, *Thinking, Fast and Slow*. New York, NY, USA: Macmillan, 2011.

[26] K. Watanabe and S. Funahashi, "Neural mechanisms of dual-task interference and cognitive capacity limitation in the prefrontal cortex," *Nature Neurosci.*, vol. 17, no. 4, pp. 601–611, 2014.

[27] E. A. Feigenbaum et al., *Computers and Thought*, vol. 7. New York, NY, USA: McGraw-Hill, 1963.

[28] C. R. Fletcher, "Understanding and solving arithmetic word problems: A computer simulation," *Behav. Res. Methods, Instruments, Comput.*, vol. 17, no. 5, pp. 565–571, 1985.

[29] J. Zhang et al., "Graph-to-tree learning for solving math word problems," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 3928–3937.

[30] Q. Wu, Q. Zhang, Z. Wei, and X.-J. Huang, "Math word problem solving with explicit numerical values," in *Proc. Joint Conf. Annu. Meeting Assoc. Comput. Linguistics, Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 5859–5869.

[31] Z. Liang et al., "MWP-BERT: Numeracy-augmented pre-training for math word problem solving," in *Proc. Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2022, pp. 997–1009.

[32] Z. Xie and S. Sun, "A goal-driven tree-structured neural model for math word problems," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 5299–5305.

[33] B. Wang et al., "Structure-unified m-tree coding solver for math word problem," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2022, pp. 8122–8132.

[34] Y. Cao et al., "A bottom-up dag structure extraction model for math word problems," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 39–46.

[35] Z. Jie, J. Li, and W. Lu, "Learning to reason deductively: Math word problem solving as complex relation extraction," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2022, pp. 5944–5955.

[36] J. Shen et al., "Generate & rank: A multi-task framework for math word problems," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2021, pp. 2269–2279.

[37] Z. Yang et al., "LogicSolver: Towards interpretable math word problem solving with logical prompt-enhanced learning," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2022, pp. 1–13.

[38] M. Zhang, Z. Wang, Z. Yang, W. Feng, and A. Lan, "Interpretable math word problem solution generation via step-by-step planning," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2023, pp. 6858–6877.

[39] X. Zhu et al., "Solving math word problem via cooperative reasoning induced language models," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2023, pp. 4471–4485.

[40] S. Imani, L. Du, and H. Shrivastava, "MathPrompter: Mathematical reasoning using large language models," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2023, pp. 37–42.

[41] H. Poon et al., "Precision health in the age of large language models," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2023, pp. 5825–5826.

[42] K. Gan and T. Wei, "Erasing the bias: Fine-tuning foundation models for semi-supervised learning," in *Proc. Int. Conf. Mach. Learn.*, 2024, pp. 14453–14470.

[43] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 22199–22213.

[44] S. Yao et al., "Tree of thoughts: Deliberate problem solving with large language models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, Art. no. 517.

[45] M. Besta et al., "Graph of thoughts: Solving elaborate problems with large language models," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 17682–17690.

[46] K. Yang et al., "If LLM is the wizard, then code is the wand: A survey on how code empowers large language models to serve as intelligent agents," 2024, *arXiv:2401.00812*.

[47] X. Wang et al., "Self-consistency improves chain of thought reasoning in language models," in *Proc. 11th Int. Conf. Learn. Representations*, 2022.

[48] Y. Zhang et al., "Cumulative reasoning with large language models," 2023, *arXiv:2308.04371*.

[49] A. Lewkowycz et al., "Solving quantitative reasoning problems with language models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 3843–3857.

[50] H. Lightman et al., "Let's verify step by step," in *Proc. 12th Int. Conf. Learn. Representations*, 2023.

[51] Y. Ding et al., "A survey on rag meets LLMs: Towards retrieval-augmented large language models," 2024, *arXiv:2405.06211*.

[52] Z. Jiang et al., "Active retrieval augmented generation," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2023, pp. 7969–7992.

[53] O. Ram et al., "In-context retrieval-augmented language models," *Trans. Assoc. Comput. Linguistics*, vol. 11, pp. 1316–1331, 2023.

[54] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 9459–9474.

[55] J. Ye, Z. Wu, J. Feng, T. Yu, and L. Kong, "Compositional exemplars for in-context learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2023, pp. 39818–39833.

[56] W. Yu, Z. Zhang, Z. Liang, M. Jiang, and A. Sabharwal, "Improving language models via plug-and-play retrieval feedback," 2023, *arXiv:2305.14002*.

[57] Y. Zhang et al., "Merging generated and retrieved knowledge for open-domain QA," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2023, pp. 4710–4728.

[58] S. Borgeaud et al., "Improving language models by retrieving from trillions of tokens," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2022, pp. 2206–2240.

[59] Z. Du et al., "CogKR: Cognitive graph for multi-hop knowledge reasoning," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 2, pp. 1283–1295, Feb. 2023.

[60] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2019, pp. 4171–4186.

[61] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 5485–5551, 2020.

[62] Y. Lan, G. He, J. Jiang, J. Jiang, W. X. Zhao, and J.-R. Wen, "Complex knowledge base question answering: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 11, pp. 11196–11215, Nov. 2023.

[63] N. Xu, Y. Gao, A.-A. Liu, H. Tian, and Y. Zhang, "Multi-modal validation and domain interaction learning for knowledge-based visual question answering," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 6628–6640, Nov. 2024.

[64] L. Yang, H. Chen, Z. Li, X. Ding, and X. Wu, "Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3091–3110, Jul. 2024.
[65] Z. Dong, Q. Dong, and C. Hao, "Hownet and the computation of meaning," in *Proc. Coling 2010: Demonstrations*, 2010, pp. 53–56.
[66] R. Speer, J. Chin, and C. Havasi, "ConceptNet 5.5: An open multilingual graph of general knowledge," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 4444–4451.
[67] J. Liu, Z. Huang, X. Lin, Q. Liu, J. Ma, and E. Chen, "A cognitive solver with autonomously knowledge learning for reasoning mathematical answers," in *Proc. IEEE Int. Conf. Data Mining*, 2022, pp. 269–278.
[68] L. Dou et al., "Towards knowledge-intensive text-to-SQL semantic parsing with formulaic knowledge," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2022, pp. 5240–5253.
[69] M. Henningsen and M. K. Stein, "Mathematical tasks and student cognition: Classroom-based factors that support and inhibit high-level mathematical thinking and reasoning," *J. Res. Math. Educ.*, vol. 28, no. 5, pp. 524–549, 1997.
[70] L. Von Rueden et al., "Informed machine learning–A taxonomy and survey of integrating prior knowledge into learning systems," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 1, pp. 614–633, Jan. 2023.
[71] M. Qu, X. Ren, and J. Han, "Automatic synonym discovery with knowledge bases," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2017, pp. 997–1005.
[72] X. Xue and Y. Wang, "Using memetic algorithm for instance coreference resolution," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 2, pp. 580–591, Feb. 2016.
[73] Q. Wu, Q. Zhang, J. Fu, and X.-J. Huang, "A knowledge-aware sequence-to-tree network for math word problem solving," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2020, pp. 7137–7146.
[74] Z. Li et al., "Seeking patterns, not just memorizing procedures: Contrastive learning for solving math word problems," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2022, pp. 2486–2496.
[75] W. Chen et al., "Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks," *Trans. Mach. Learn. Res.*, vol. 2023, 2023.
[76] L. Gao et al., "PAL: Program-aided language models," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2023, pp. 10764–10799.
[77] Z. Zhou, Y. Wang, X. Xie, L. Chen, and C. Zhu, "Foresee urban sparse traffic accidents: A spatiotemporal multi-granularity perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3786–3799, Aug. 2022.
[78] J. Wu, L. Yang, M. Okumura, and Y. Zhang, "MRKE: The multi-hop reasoning evaluation of LLMs by knowledge edition," 2024, *arXiv:2402.11924*.
[79] K. Cobbe et al., "Training verifiers to solve math word problems," 2021, *arXiv:2110.14168*.
[80] DeepSeek-AI, "Deepseek-v3 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2412.19437
[81] T. Xiao et al., "Learning to solve geometry problems via simulating human dual-reasoning process," in *Proc. Int. Joint Conf. Artif. Intell.*, 2024, pp. 6559–6568.
[82] L. Wu et al., "Supporting your idea reasonably: A knowledge-aware topic reasoning strategy for citation recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 8, pp. 4275–4289, Aug. 2024.
[83] H. Pei, B. Yang, J. Liu, and K. C.-C. Chang, "Active surveillance via group sparse Bayesian learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 3, pp. 1133–1148, Mar. 2022.
[84] C. Zhao, H. Zhao, X. Li, M. He, J. Wang, and J. Fan, "Cross-domain recommendation via progressive structural alignment," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 6, pp. 2401–2415, Jun. 2024.
[85] R. Bommasani et al., "On the opportunities and risks of foundation models," 2021, *arXiv:2108.07258*.
[86] M. Burtsev, M. Reeves, and A. Job, "The working limitations of large language models," *MIT Sloan Manage. Rev.*, vol. 65, no. 1, pp. 1–5, 2023.
[87] L. Yang et al., "Buffer of thoughts: Thought-augmented reasoning with large language models," 2024, *arXiv:2406.04271*.
[88] A. Davies et al., "Advancing mathematics by guiding human intuition with AI," *Nature*, vol. 600, no. 7887, pp. 70–74, 2021.
[89] J. Sourati and J. A. Evans, "Accelerating science with human-aware artificial intelligence," *Nature Hum. Behav.*, vol. 7, no. 10, pp. 1682–1696, 2023.

Jiayu Liu received the BS degree in applied mathematics from the University of Science and Technology of China (USTC), in 2020. He is currently working toward the PhD degree with the School of Data Science, USTC. His research interests include data mining, knowledge representation and learning, and mathematical reasoning. He has published papers in refereed conference proceedings, such as NeurIPS 2024, KDD 2023, AAAI 2023, and ICDM 2022.

Zhenya Huang (Member, IEEE) received the PhD degree from the University of Science and Technology of China (USTC), in 2020. He is currently an associate professor with USTC. His main research interests include data mining, knowledge reasoning, nature language processing, and intelligent education. He has published more than 50 papers in refereed journals and conference proceedings, including *IEEE Transactions on Knowledge and Data Engineering*, *ACM Transactions on Information Systems*, *IEEE Transactions on Neural Networks and Learning Systems*, AAAI, KDD, SIGIR, and ICDM. He has served regularly on the program committee of numerous conferences and is a reviewer for the leading academic journals.

Qi Liu (Member, IEEE) received the PhD degree from the University of Science and Technology of China (USTC), in 2013. He is currently a professor with USTC. His general research areas include data mining and knowledge discovery, and artificial intelligence. His research is supported by the National Science Fund for Excellent Young Scholars and the Youth Innovation Promotion Association of Chinese Academy of Sciences. He has published more than 100 papers in refereed journals and conference proceedings, such as *IEEE Transactions on Knowledge and Data Engineering*, *ACM Transactions on Information Systems*, *IEEE Transactions on Neural Networks and Learning Systems*, NeurIPS, ICML, ICLR, AAAI, and KDD. He has served regularly in the program committee of numerous conferences and is a reviewer for the leading academic journals. He is the recipient of the KDD 2018 Best Student Paper Award (Research) and the ICDM 2011 Best Research Paper Award, and the Alibaba DAMO Academy Young fellow.

Zhiyuan Ma received the BS degree in computer science and technology from Shandong University, in 2023. He is currently working toward the first-year master's degree with the School of Computer Science and Technology, USTC. His research interests include knowledge representation and mathematical reasoning. He has published papers in refereed journals and conference proceedings, such as JUSTC and KDD 2023.

Chengxiang Zhai received the PhD degree in computer science from Nanjing University, in 1990, and the PhD degree in language and information technologies from Carnegie Mellon University, in 2002. He is a professor of the Department of Computer Science, University of Illinois at Urbana-Champaign (UIUC), Champaign, Illinois, where he also holds a joint appointment with the Carl R. Woese Institute for Genomic Biology, Department of Statistics, and the School of Information Sciences. His research interests are in the general area of intelligent information systems, including specifically information retrieval, data mining, natural language processing, machine learning, and their applications in domains such as biomedical informatics, and intelligent education systems. He served as an associate editor of journals in multiple areas, including *ACM Transactions on Information Systems*, *Information Processing and Management*, and *ACM Transactions on Knowledge Discovery from Data*. He is a program co-chair of CIKM-2004, SIGIR-2009, and WWW-2015. He is a general conference co-chair of CIKM-2016, WSDM-2018, and IEEE BigData-2020. He is an ACM fellow and a member of the ACM SIGIR Academy, and received a number of awards, including ACM SIGIR Gerard Salton Award, multiple best paper awards such as the ACM SIGIR 2004 Best Paper Award, and the ACM SIGIR Test of Time Award (three times).

Enhong Chen (Fellow, IEEE) received the PhD degree from the University of Science and Technology of China (USTC), in 1996. He is currently a professor of USTC. His research areas include data mining and knowledge discovery, machine learning and artificial intelligence. His research is supported by the National Science Foundation for Distinguished Young Scholars of China. He has published more than 200 papers in refereed conferences and journals, including *IEEE Transactions on Knowledge and Data Engineering*, *ACM Transactions on Information Systems*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *IEEE Transactions on Neural Networks and Learning Systems*, ICML, NeurIPS, KDD, ICLR and AAAI. He is an associate editor of *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Systems, Man, and Cybernetics*, *ACM Transactions on Intelligent Systems and Technology*, *World Wide Web Journal*. He has served regularly on the organization and program committees of numerous conferences, including as a program co-chair of ICKG-2020, a program co-chair for PAKDD-2022. He received the Best Application Paper Award on KDD 2008, the Best Student Paper Award on KDD 2018 (Research), the Best Research Paper Award on ICDM 2011.