Enhancing Mathematical Reasoning Through Autonomously Learning Knowledge

Jiayu Liu, Zhenya Huang, *Associate Member, IEEE*, Enhong Chen, *Fellow, IEEE*, Qi Liu, *Member, IEEE*, Hongke Zhao, Xin Lin, Jing Sha, and Shijin Wang

***Abstract*—Enabling machines to solve mathematical problems is a vital endeavor in developing intelligence that emulates human-like thinking and reasoning. However, most existing approaches focus on reconstructing human comprehension of problems, which are still far from enough since they neglect the fundamental human ability to learn knowledge from experiences. In this article, we focus on empowering models with the cognitive capacity to autonomously learn knowledge from mathematical problem-solving. We first propose a Cognitive Solver (CogSolver) that contains an intelligent *BRAIN-ARM* framework as the cognitive structure and operates the knowledge learning process in *Store-Apply-Update* steps inspired by two cognitive science theories. The *BRAIN* system *stores* three basic types of mathematical knowledge, and the *ARM* system *applies* them organically in answer reasoning process. After solving problems, the *BRAIN updates* its stored knowledge based on the *ARM*'s feedback, with knowledge filters to eliminate redundancies and foster a more rational knowledge base. Our CogSolver carries out the above three steps iteratively, emulating a more human-like behavior. Furthermore, in order to overcome knowledge forgetting during the learning process, we extend CogSolver to CogSolver+ by incorporating an essential knowledge *Recall* mechanism, which is inspired by another prominent cognitive theory. We first discuss and fuse three crucial factors in simulating human memory replay. Then, we propose a influenced-based method with a theoretical guarantee of efficiency to consolidate the *updated* knowledge. Experiments on three math word problem benchmarks demonstrate the improvements of our CogSolver and CogSolver+ in answer reasoning and clearly illustrate how they acquire knowledge, leading to superior interpretability.**

***Index Terms*—Knowledge learning, mathematical reasoning, cognitive behaviors, human-inspired model.**

I. Introduction

AUTOMATICALLY solving mathematical problems constitutes a pivotal step in the development of general artificial intelligence. It necessitates machine understanding of natural language, acquisition of mathematical knowledge, and execution of human-like logical thinking. Consequently, the capacity of mathematical reasoning serves as an indicator of the attained intelligent level [1]. In this article, we focus on a fundamental category of problems: math word problems (MWP), which is an important branch that has attracted much attention since the 1960s [2]. As shown in Fig. 1, the MWP can be expressed as a short narrative that describes a problem and raises a question about an unknown quantity. The model is required to comprehend the verbal description ("Jack has 3...") that involves words (e.g., "Jack") and quantities (e.g., 3), and reason a mathematical expression (e.g., $3 + 2$) to get the answer (e.g., 5).

In the literature, existing efforts in solving MWP can be categorized into rule-based, statistic-based, semantics parsing-based, and deep learning-based [1], [3]. Recently, inspired by researches on natural language processing, the Sequence-to-Sequence (Seq2Seq) method [2], [4], [5] has emerged, which inputs the problem sentence and outputs the expression as a sequence directly. More advanced work, including Seq2Tree [6], Graph2Tree [7], and Seq2DAG [8], develop to enhance the validity of expressions by decoding the problem as a tree or a Direct Acyclic Graph (DAG). Though previous work has achieved great success, there remains a disparity between these approaches with human-like intelligence, manifesting in two primary aspects. On one hand, humans can naturally learn knowledge from solving mathematical problems [9], e.g., we acquire the knowledge that "banana" (or "apple") is a kind of "fruit" when solving the problem illustrated in Fig. 1. This knowledge is highly interpretable to humans and can be expressed explicitly. Nevertheless, existing methods mainly focus on training models to improve the comprehension ability [10], [11], [12], such as better understanding the semantic meaning and sentence structure in problems. Their learning results are

Received 18 August 2023; revised 24 August 2024; accepted 11 November 2025. Date of publication 19 November 2025; date of current version 4 February 2026. The work of Zhenya Huang was supported by Young Elite Scientists Sponsorship Program by CAST under Grant 2024QNRC001. This work was supported in part by the National Natural Science Foundation of China under Grant U23A20319, Grant 62477044, Grant 72471165, Grant 62337001, and Grant 62525606 and in part by the Fundamental Research Funds for the Central Universities under Grant WK2150110038. Recommended for acceptance by M. Cheng. *(Corresponding author: Enhong Chen.)*

Jiayu Liu and Qi Liu are with the School of Artificial Intelligence and Data Science, University of Science and Technology of China, Hefei 230026, China, and also with the State Key Laboratory of Cognitive Intelligence, Hefei 230000, China (e-mail: jy251198@mail.ustc.edu.cn; qiliuql@ustc.edu.cn).

Zhenya Huang, Enhong Chen, and Xin Lin are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China, and also with the State Key Laboratory of Cognitive Intelligence, Hefei 230000, China (e-mail: huangzhy@ustc.edu.cn; cheneh@ustc.edu.cn; linx@mail.ustc.edu.cn).

Hongke Zhao is with the College of Management and Economics, Laboratory of Computation and Analytics of Complex Management Systems (CACMS), Tianjin University, Tianjin 300072, China (e-mail: hongke@tju.edu.cn).

Jing Sha and Shijin Wang are with the iFLYTEK Research and State Key Laboratory of Cognitive Intelligence, Hefei 230000, China (e-mail: jingsha@iflytek.com; sjwang3@iflytek.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2025.3634507, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2025.3634507

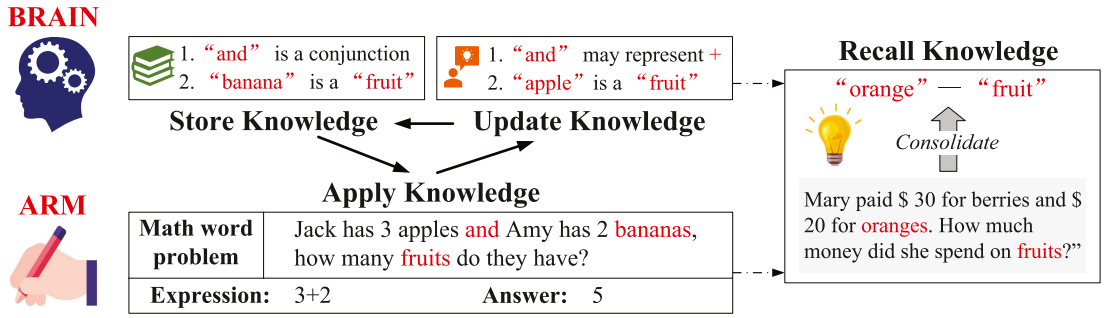


Fig. 1. Example of human cognitive structure and learning process for math word problems (MWP).

often wrapped in parameters of neural networks, which lack the interpretability for the aforementioned relational knowledge. On the other hand, humans possess the ability to apply the learned knowledge to address unseen problems [13], [14]. For example, in Fig. 1, the learned knowledge ("banana" is a kind of "fruit") is the basis for correctly reasoning the expression $2 + 4$ to another problem "John has 2 bananas and Lisa has 4 pears, how many fruits do they have?". Such ability is crucial for constructing a model with high generalization performance. In summary, we argue that it is essential to empower models with the capability to autonomously learn and apply knowledge as human cognitive processes operate [15], [16], which we specifically explore under MWP scenario in this article.

To this end, we turn to some cognitive science theories for inspiration. We draw the first insight from dual process theory [17], [18], [19], which posits that human cognitive structure consists of two systems: System 1 and System 2. System 1 is a fast, heuristic system that retrieves and provides information, while System 2 is a slow, analytical reasoning system that solves tasks based on the information provided by System 1. Therefore, we basically conceive a solver comprising *BRAIN-ARM* systems to simulate the structure of human cognition. The *BRAIN* system functions similarly to System 1, which mainly handles knowledge processing, such as acquiring and providing knowledge. The *ARM* system, akin to System 2, aims to analytically apply knowledge to reason the answer step by step.

One step further, inspired by information processing theory [20], [21], [22], we operate the knowledge learning process as a cyclical procedure consisting of three iterative steps: *Store-Apply-Update*. To be more specific, in Fig. 1, a human first *stores* the knowledge that "and" is a conjunction and "banana" is a kind of "fruit". Then she *applies* this knowledge to answer the problem "Jack has...". After figuring out this problem, she might *update* the knowledge by realizing that "and" may also represent "+" and "apple" is another kind of "fruit", which is useful for answering other problems. The above three steps are performed in the two systems respectively, where *BRAIN* governs the *storage* and *update* steps, and *ARM* conducts the *application* step.

However, it is nontrivial to carry out these three steps. First, there are various types of knowledge in MWP, such as the semantics of tokens (words and operators), relationships between tokens (e.g., "apple"-"fruit", "add"-"+"), as well as mathematical properties (e.g., commutative law). It is challenging to represent and store diverse knowledge altogether. Second, we need to devise specialized mechanisms for applying different types of knowledge while also integrating contextual information during the reasoning process. For instance, while "and" implies "+" within the problem "Jack has..." in Fig. 1, it may represent as a conjunction in other sentences. Thus, knowledge about the relationship between "and"-"+" should be applied differently under different context. Third, the mechanism of knowledge updating in human brain remains underexplored at present. It is coupled with the manner of knowledge storage, which makes it more challenging to design suitable update methods.

To address these challenges, in our preliminary work [23], we proposed a cognitive solver (CogSolver) that integrated knowledge learning and mathematical reasoning. Specifically, in CogSolver, we first defined and stored three types of essential mathematical knowledge in *BRAIN*, including *semantics knowledge*, *relation knowledge*, and *mathematic rule knowledge*. Then, we proposed a knowledge-aware module and a commutative module to apple this knowledge for enhancing the reasoning ability of *ARM*. After solving the problem, *BRAIN* updated the *semantics knowledge* through back-propagation and *relation knowledge* with knowledge filters to reduce redundancy. The updated knowledge was further stored, and the cycle of *Store-Apply-Update* repeated in the subsequent learning process to gradually form a mathematical knowledge base in *BRAIN*.

In CogSolver, the *Store-Apply-Update* steps have achieved a basic realization of how humans autonomously and continuously acquire knowledge from problem solving. However, during the iterations of these steps, the *BRAIN-ARM* systems may lost its grasp of part of knowledge (e.g., "orange"-"fruit") to some extent. This is because in each iteration, CogSolver only focuses on the problems solved within the current *apply* step and on this basis *updates* knowledge, which possibly overwrite the experience (i.e., the already learned knowledge) accumulated from previous iterations, particularly those garnered at earlier stages. To address this dilemma between updating and preserving existing knowledge, we further resort to the complementary learning system theory [24], [25], [26], which points out that there exist specialized memory replaying in human cognitive systems to consolidate learned reasonable knowledge. For example, a human could recall another problem "Mary paid... $20 for oranges.... spend on fruits?" after completing the problem "Jack

has..." in Fig. 1, because it also examines the knowledge related to "fruit". By applying the updated knowledge to solve it, she can *recall* the relationship between "orange" and "fruit" and mitigate the weakening of it. According to the theory, this process is conducted by the collaboration of hippocampus and neocortex within the human brain [27], in which the hippocampus replays past experiences (i.e., solved problems) to the neocortex and the neocortex reinforces the long-term knowledge [26], [27], [28].

Building upon the above understanding, in this article, we extend CogSolver to CogSolver+ by introducing a necessary knowledge *Recall* mechanism in *BRAIN-ARM* systems, which consolidates the learned knowledge after each iteration of *Store-Apply-Update* steps in a "Replay-Consolidate" manner. Specifically, in CogSolver+, we first propose a problem replay module to retrieve problems solved in previous knowledge *apply* steps as experiences. We consider the impacts of three essential pedagogical and psychological factors on memorization, including knowledge relevance, problem relevance, and temporal relevance, and fuse them together to obtain comprehensive replayed results. Then, to emulate how humans consolidate reasonable long-term knowledge, we introduce an intuitive objective to assess the generalization performance of the updated knowledge on the replayed problems. Nevertheless, estimating this objective necessitates additional parameter optimization, and we theoretically demonstrate that it entails prohibitively expensive time complexity with respect to the numbers of knowledge and parameters. Given the infeasibility in reality, we formally derive an approximate estimation utilizing the influence function and combine it with stochastic estimation to reduce computational complexity. Based on this approximation, we design momentum-based methods to consolidate (either strength or weaken) each segment of the updated knowledge specifically.

We conduct experiments on two real-world datasets to evaluate both CogSolver and CogSolver+. Our experimental results show that they can effectively learn reasonable knowledge from MWP and further enhance mathematical intelligence with interpretable reasoning processes. In particular, our knowledge *Recall* mechanism in CogSolver+ demonstrates remarkable efficiency and effectiveness in consolidating reasonable long-term knowledge during learning. To the best of our knowledge, this work is one of the first few attempts to autonomously learn knowledge for solving MWP by constructing *BRAIN-ARM* systems and specifying the learning process into *Store-Apply-Update* steps and knowledge *Recall* mechanism, whose main ideas are quite general and can be potentially applicable to other mathematical problems and reasoning tasks.

The remainder of this article is organized as follows. In Section II, we review the related works on MWP and cognitive science theories. Then, we give the problem definition in Section III. In Section IV and Section V, we introduce our CogSolver and CogSolver+, respectively. In Section VI, we report the experimental results on three representative datasets. Finally, we conclude the article in Section VII.

II. Related Work

In this section, we summarize the related work as follows.

Math Word Problems: Mathematical reasoning endeavors to acquire knowledge from data and utilize it in resolving mathematical problems. Among various problem categories, MWP represents a foundational task that has spurred extensive research in the premier venues of artificial intelligence since the 1960s [1], [29], [30]. In the literature, existing MWP solvers can be classified into rule-based, statistic-based, semantics parsing-based, and deep learning-based [1]. Specifically, rule-based methods rely on manually crafted schemas and pattern matching [31], while statistic-based methods leverage traditional machine learning models like SVM to select and complete predefined templates [32]. Semantic parsing-based methods focus on the semantic structure of the textual sentences and derive math expressions after constructing them into logic forms [33]. However, these methods require hand-crafted templates, rules, and representation language, which limits their applications on large-scale datasets and causes low generality. With the development of deep learning, Wang et al. [4] first used a recurrent neural network to translate math word problems into equations. Based on it, many advanced reasoning manners have been further developed, such as Seq2Tree [6], [11], [34], Graph2Tree [7], [35], deductive reasoning [36], and reinforcement learning [37], with many models emerging from each of them. For example, Xie et al. [6] proposed a top-down goal decomposition process to generate expression trees, motivated by the goal-driven mechanism in human problem solving. Zhang et al. [7] constructed graphs to enrich quantities' representations with the relationships and order information. Wu et al. [38] incorporated commonsense and built entity graph to obtain better problem. Liu et al. [39] and Yang et al. [12] introduced mathematical formulas, developing sophisticated formula-guided reasoning mechanisms and formula prompts for problem-solving, respectively. Besides, pre-trained language models such as BERT [40], RoBERTa [41], BART [42] have also been utilized to promote comprehension of problems. In the context of large language models (LLMs), research has primarily focused on using them (e.g., GPT-J) to generate natural language descriptions of reasoning paths [43], [44] or to invoke code to solve problems [45]. Apart from arithmetic problems, other mathematical problems like equation set problems [8], geometry problems [46], and conjectures discovery [47] also attract great attention.

Cognitive Science Theories: Cognitive science investigates the human intelligence by bringing studies in psychology, linguistics, anthropology, philosophy, and so on [48]. Here, we introduce three theories about human cognitive structure, reasoning process and mechanisms relevant to our work. First, *dual process theory* [17], [18], [19] points out that two separate cognitive systems consist of human thinking and reasoning structure. System 1 includes instinctive behaviors that are innately programmed and retrieves information in a fast, unconscious, and implicit way. System 2 conducts a slow, conscious, and explicit sequential thinking process. Based on this, previous attempts have been made for some machine learning tasks, such as question answering [49], [50] and knowledge graph reasoning [27]. Second, *information processing theory* [20], [21], [22] states that the mind's machinery are conducted in sensory memory, short-term memory, and long-term memory, and defines cognitive

processes as mental activities that help information transfer from one memory to another, such as attention, perception, repetition, coding, and retrieving [21]. As we focus on how models gain knowledge from experience, mechanisms associated with long-term memory should be emphasized. We summarize the related activities into three steps: (1) the ability to hold information refers to knowledge storage, (2) repetition and coding are activities that transfer information from short-term memory to long-term memory, which means summarizing the commonly used methods or steps in MWP as knowledge and is thus relevant to knowledge update, (3) retrieving information from long-term to short-term is taking advantage of what already know, coincident with the process of knowledge application for answer reasoning. Third, *complementary learning system theory* [24], [25], [26] explains neuropsychological and behavioral phenomena in the domain of human recognition memory. It clarifies how the two specialized learning and memory components in human brain, i.e., hippocampus and neocortex, account for memory encoding, recall, and consolidation [51]. In brief, the hippocampus plays an crucial role in organizing and replaying episodic memories (i.e., experience) to the neocortex, and the neocortex gradually integrates these episodes to consolidate learned semanticized information (i.e., knowledge). Some researchers have investigated this concept to address catastrophic forgetting in continual graph learning [27], image classification [28], and other areas.

Our work differs from previous studies as follows. First, existing approaches mainly focus on training models to better understand the problems (e.g., incorporate additional semantic features) or master the heuristic reasoning manner (e.g., the goal-driven mechanism). Comparatively, our work investigates the principled way of how humans learn knowledge and designs methods to apply the learned knowledge for enhancing answer reasoning. Second, existing methods that introduce knowledge into MWP rely on manually constructed external knowledge bases, and thus are restricted when such knowledge is unavailable. Comparatively, our work can autonomously learn knowledge from scratch, which is more in line with the goal of general artificial intelligence and achieves better transferability. Last but not least, we construct a superior human-inspired cognitive framework by specifying two intelligent systems stated in dual process theory, unifying them with three learning steps summarized from information process theory, and developing an essential knowledge recall mechanism based on complementary learning system theory.

III. Problem Definition

The input of a math word problem P is a sequence of n word tokens and numeric values described in natural language: $P = \{p_1, \ldots, p_n\}$ (e.g., "Jack has... have?" in Fig. 1), where p_i is either a word token (e.g., "Jack") or a numeric value (e.g., "3"). In particular, we define the set of numeric values (e.g., "3","2") that appear in P as N_P. The output of P is a numeric answer s_P (e.g, "5") derived from an expression $E_P = \{y_1, \ldots, y_m\}$ (e.g, "3 + 2"). E_P is a sequence of m symbols, where each y_i comes from the decoding vocabulary V_P. V_P is composed of the operator set V_O (e.g., $\{+, \times, -, \div\}$), numeric constant set V_C (e.g., $1, 2, \pi$) and N_P, i.e., $V_P = V_O \cup V_C \cup N_P$. Note that different problems may have different V_P since N_P varies with P.

The goal of MWP is to train a model that reads the input problem P and generates a valid mathematical expression E_P, based on which calculates the numeric answer s_P.

IV. CogSolver: Cognitive Solver

In this section, we introduce our Cognitive Solver (CogSolver) that achieves the primary goal of autonomously learning and applying knowledge to solve MWP. Specifically, we first describe *BRAIN-ARM* systems that serve as the underlying cognitive structure of CogSolver (Section IV-A). Then, we specify *Store-Apply-Update* steps (Sections IV-B1-IV-B3, respectively) that simulate the concrete human cognitive processes operating in *BRAIN* and *ARM*.

A. Cognitive Structure: BRAIN-ARM Systems

We establish two systems: *BRAIN-ARM* in CogSolver as shown in Fig. 2, referring to the idea of dual process theory [17], [18], [19] to model human cognitive structure. Specifically, the top-level system is *BRAIN* that is responsible for knowledge management and processing, and the bottom-level system is *ARM* that solves specific math word problems under the guidance of *BRAIN*. They play the role of System 1 and System 2 in the theory respectively.

The autonomously learning process for MWP is as follows. Given a problem P, CogSolver first retrieves *stored* knowledge relevant to P from *BRAIN* and sends it to *ARM*. Then, *ARM applies* the knowledge and conducts reasoning to figure out the problem (i.e., generate the expression E_P and calculate s_P). Finally, CogSolver *updates* the knowledge in *BRAIN* according to the feedback of *ARM* (i.e., whether the answer is correct) for future applications. The updated results are further *stored* in *BRAIN* and these *Store-Apply-Update* steps iterate to achieve a continuous learning process. Terminally, we name such an iteration of three steps as *learning iteration* in this article.

B. Cognitive Process: Store-Apply-Update Steps

Drawing insights from information processing theory [20], [21], [22], we carry out the knowledge learning process of CogSolver in three iterative steps: *Store-Apply-Update*. They are operated in *BRAIN-ARM* systems, respectively. Specifically, the *BRAIN* conducts the knowledge *Store* and *Update* steps, and the *ARM* takes charge of the *Apply* step. Their implementation details are described as follows.

1) Knowledge Storage: In CogSolver, *BRAIN* stores three necessary types of knowledge for MWP as depicted in Fig. 2, including *semantics knowledge*, *relation knowledge*, and *mathematic rule knowledge*. They are all explicit and interpretable.

Semantics knowledge refers to the meanings of mathematical tokens (words and operators), which can be represented as feature vectors in the conceptual space [18]. We denote the semantic vectors of tokens, including words as $W = \{w_1, \ldots, w_N, w_i \in R^d\}$, and operators as $O = \{o_1, \ldots, o_C, o_c \in R^d\}$, where N, C

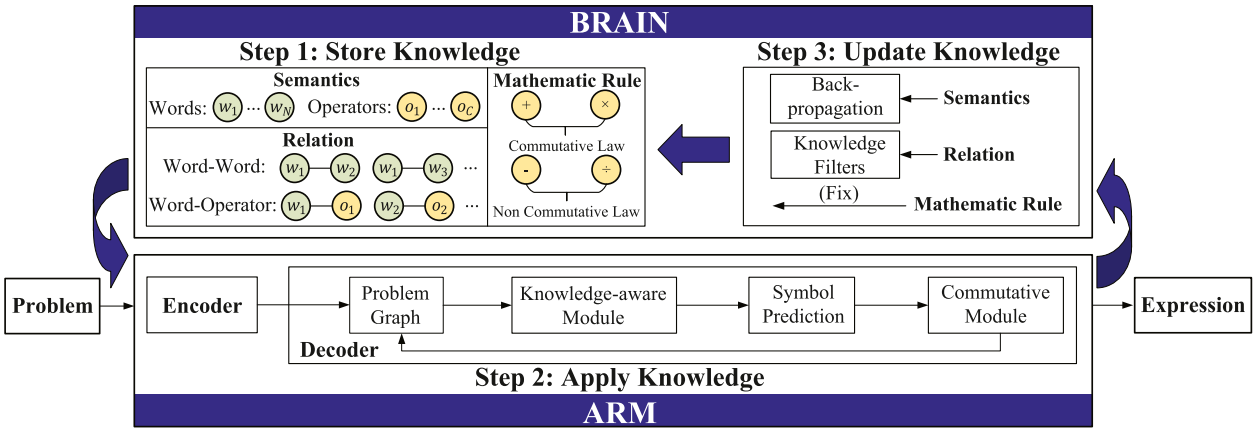


Fig. 2. Framework of CogSolver. The top-level *BRAIN* stores and updates knowledge. The bottom-level *ARM* applies knowledge to solve MWP.

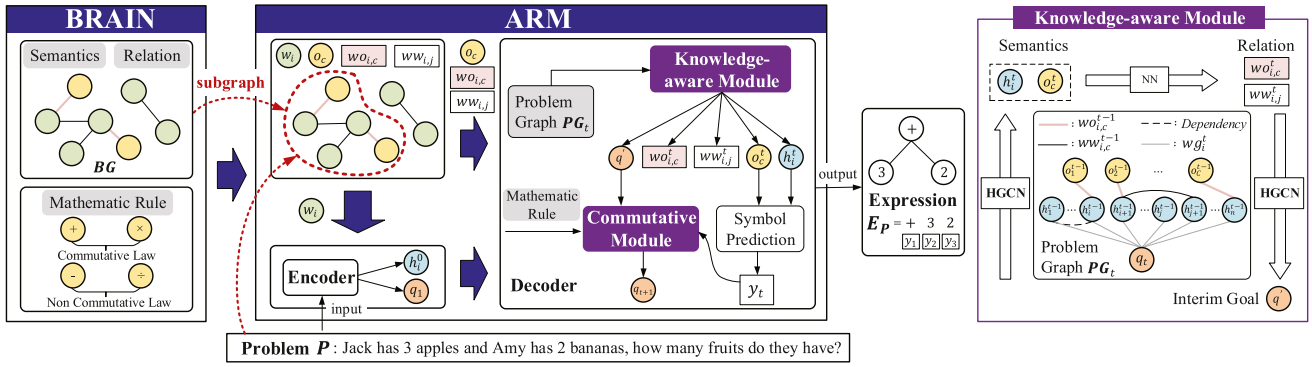


Fig. 3. Knowledge application. Left part shows the problem solving process. Right part illustrates details of knowledge-aware module with hierarchical graph convolutional network (HGCN).

are the number of words and operators in the field of MWP, and d is the dimension.

Relation knowledge describes the relationships among mathematical tokens, which is further divided into two types. The first is word-word relation or the so-called commonsense [52]. Notice that it can be generalized to encompass specific meanings (e.g., hypernymy [53]) or conform to particular structures that can be used to learn knowledge by powerful graph learning models [54]. In this article, we focus more on the type of knowledge rather than its particular semantics or structures. Therefore, we adopt a real value $ww_{i,j} \in [0, 1]$ to store the overall relation strength between words i and j instead of a binary one, because one word can have multiple meanings, not all of which have the relationship. The second is word-operator relation (e.g., "add" is highly related to "+"), which generally captures the logical correspondence between words and operators. Similarly, we denote the relation strength between word i and operator c as $wo_{i,c} \in [0, 1]$.

Mathematic rule knowledge refers to several important mathematical properties, such as commutative law, associative law, and distributive law. In this study, we primarily consider the commutative law for its simplicity and representativeness [8]. Following its mathematical property, we set the rule in *BRAIN* that "+" and "×" satisfy the commutative law, while the others (e.g., "−", "÷") do not. This setup resembles a teacher directly introducing the commutative law to students, which can avoid confusion of basic mathematical concepts.

Specifically, the *semantics knowledge* and *relation knowledge* together constitute a knowledge graph in *BRAIN*, denoted as $BG = (V, E)$, where the former (i.e., words and operators) serve as the nodes $V = \{w_i, o_c | i = 1, \ldots, N, c = 1, \ldots, C\}$ and the latter (i.e., word-word/operator relation) serve as the edges $E = \{ww_{i,j}, wo_{i,c} | i, j = 1, \ldots, N, c = 1, \ldots, C\}$. In BG, node representations and edge strength are initialized randomly and will be updated as illustrated in Section IV-B3 to achieve learning from scratch autonomously.

2) Knowledge Application: The process of solving a specific problem is completed by the *Apply* step conducted in *ARM*. Specifically, as shown in Fig. 3, it follows the encoder-decoder manner, wherein we propose a novel knowledge-aware module and a commutative module to augment *ARM*'s capacity of answer reasoning based on knowledge sourced from *BRAIN*. Generally, given a problem $P = \{p_1, \ldots, p_n\}$, we first retrieve relevant *semantics knowledge* into *ARM*'s encoder to attain problem understanding. Then, we combine it with *relation knowledge* to construct a problem-specific graph PG_t that organizes

the contextual information and necessary knowledge in *ARM*'s decoder. After that, we conduct cognitive reasoning on PG_t $(t = 1, \ldots, m)$ in the knowledge-aware module to predict the expression $E_P = \{y_1, \ldots, y_m\}$ (represented by tree) symbol by symbol, during which the *mathematic rule knowledge* is applied in the commutative module to improve the reasoning results implicitly. In the following, we will first introduce the architecture of encoder and decoder in *ARM* and then explain the technical details of decoder.

Encoder. *ARM*'s encoder reads problem P and produces hidden representations of words $H = (h_1^0, \ldots, h_n^0)$ as well as the initial reasoning goal q_1 for decoder. Specifically, we initialize word p_i in P with *semantics knowledge* w_i from *BRAIN* and feed $\{w_1, \ldots, w_n\}$ into a neural network f_θ presented in (1). Note that f_θ can be implemented ranging from LSTM, BERT, to specific MWP encoders. We do not emphasize their differences and adopt the non-pre-trained encoder HMS [11] (since it simulates human reading habits) and the pre-trained BERT [55] (due to its superiority in text mining) in experiments to validate the generality of our CogSolver (please see Section VI for details).

$$H \in \mathbb{R}^{n \times d_1}, q_1 \in \mathbb{R}^{d_1} = f_\theta(w_1, \ldots, w_n). \tag{1}$$

Decoder. *ARM*'s decoder employs the basic goal-driven mechanism [6] to reason expression E_P. It generates one token y_t at time t based on the problem representation H, the current reasoning goal q_t, and all previous outputs $\{y_1, \ldots, y_{t-1}\}$. In general, given problem P and the target expression E_P in the training set, the prediction loss is:

$$Loss_P = \sum\nolimits_{t=1}^{m} -\log \mathbf{P}(y_t \mid y_1, \ldots, y_{t-1}, H, q_t). \tag{2}$$

As shown in Fig. 3, our decoder consists of four main parts: **Problem Graph**, **Knowledge-aware Module**, **Symbol Prediction**, and **Commutative Module**. Generating each y_t requires a process executed in these four parts. Specifically, at time t $(t = 1, \ldots, m)$, the decoder first constructs a problem graph PG_t that organizes relevant *semantics knowledge* and *relation knowledge* retrieved from *BRAIN* as well as the reasoning goal q_t. Then, the knowledge-aware module calculates context-related semantics and relation in PG_t and derives an interim goal q' as the intermediate state of the current reasoning step. Based on the context-related semantics in PG_t, a pointer-generator network [37] is used to predict symbol y_t. Finally, the commutative module applies the *mathematic rule knowledge* to generate the next reasoning goal q_{t+1} based on q', and the problem graph PG_t evolves into PG_{t+1}, which is used to conduct the next reasoning step for generating y_{t+1}.

In particular, to apply the *mathematic rule knowledge* (i.e., commutative law), we introduce a commutative loss $Loss_C$ in the commutative module to enforce that the reasoning results conform to the commutative law. Overall, combined with (2), the loss to be minimized is:

$$Loss = Loss_P + \lambda \cdot Loss_C, \tag{3}$$

where λ is a hyper-parameter that keeps a balance between symbol prediction loss and commutative loss. Below, we present the details of the four parts in *ARM*'s decoder.

Problem Graph: Given problem P, we maintain a problem graph $PG_t = (V_t, E_t)$ to manage relevant *semantics knowledge* and *relation knowledge*. V_t is a node set composed of words $\{h_i^{t-1}, i = 1, \ldots, n\}$ from P, all operators $\{o_c^{t-1}, c = 1, \ldots, C\}$, and current goal q_t. E_t is a set of undirected weighted edges among words, operators, and the goal. We denote the knowledge edges between words and words, words and operators as $ww_{i,j}^{t-1}, wo_{i,c}^{t-1}$, respectively. We also incorporate dependency edges between words in a clause by *stanford corenlp toolkit* [56] to capture the semantic structure of problem P. Besides, we build an edge wg_i^t between goal q_t and word i by the hierarchical attention mechanism [11], which can be used to reflect how focused q_t is on i.

In summary, PG_t is composed of three types of nodes and four types of edges. The superscript $t-1$ indicates that $h_i^{t-1}, o_c^{t-1}, ww_{i,j}^{t-1}, wo_{i,c}^{t-1}$ are calculated before step t. To be notice, at the first step $t = 1$, $o_c^0, ww_{i,j}^0, wo_{i,c}^0$ in PG_1 are initialized with the *semantics knowledge* o_c and *relation knowledge* $ww_{i,j}, wo_{i,c}$ retrieved from *BRAIN*. Thus, PG_t can be seen as a subgraph of the overall BG stored in *BRAIN* that is fine-tuned for current reasoning step.

Knowledge-aware Module: Directly applying knowledge including semantics h_i^{t-1}, o_c^{t-1} and relation $ww_{i,j}^{t-1}, wo_{i,c}^{t-1}$ in PG_t may be unsuitable for current reasoning step (recall the example of "and" representing different meanings in different problem contexts in Fig. 1). Thus, this module aims at integrating the knowledge with contextual information implied in current goal q_t for subsequent symbol prediction. It processes semantics, relation and derives an interim goal successively, illustrated in (K1.), (K2.) and (K3.) respectively.

(K1.) For semantics in PG_t, we propose a hierarchical graph convolutional network (HGCN) to propagate the information from the current goal q_t to the upper levels of PG_t, which is inspired by human reasoning habits. For example, in Fig. 3, to accomplish the goal q_t that represents "How many fruits do they have", it is natural to first notice the word "and" and then infer the symbol "+" based on its relationship with "and". It reflects that the contextual information in q_t should be first propagated from the reasoning goal to words, and then to operators, following a hierarchical manner [57]. To achieve this process, in HGCN, we first update the word semantics h_i^t by:

$$\begin{aligned} AGG_i &= \left[wg_i^t \cdot q_t, \sum\nolimits_{j=1}^{n} ww_{i,j}^{t-1} \cdot h_j^{t-1}, \sum\nolimits_{j \in \mathcal{N}_d(i)} h_j^{t-1} \right], \\ h_i' &= ReLU(\mathbf{W}_u \cdot AGG_i + b_u), \\ f_i &= \sigma(\mathbf{W}_{fw} \cdot [h_i^{t-1}, AGG_i] + b_{fw}), \\ h_i^t &= f_i \cdot h_i^{t-1} + (1 - f_i) \cdot h_i', \end{aligned} \tag{4}$$

where $\sigma(\cdot)$ is the sigmoid function, $\mathbf{W}_*, b_*$ are learnable parameters. AGG_i aggregates word i's neighbors in different relationships with concatenation $[\cdot]$ ($\mathcal{N}_d(i)$ are neighbors with dependency edges to i), except its upper-level neighbors (operators). The forget gate f_i controls how much contextual information from the previous step $t-1$ is retained. Then, the context-related operator semantics o_c^t is calculated similarly

based on c's neighbors (i.e., all words $\{h_i^t\}$) by:

$$\begin{aligned} o_c' &= ReLU(\mathbf{W}_o \cdot \sum\nolimits_{i=1}^{n} wo_{i,c}^{t-1} \cdot h_i^t + b_o), \\ f_c &= \sigma(\mathbf{W}_{fo} \cdot [o_c^{t-1}, \sum\nolimits_{i=1}^{n} wo_{i,c}^{t-1} \cdot h_i^t] + b_{fo}), \\ o_c^t &= f_c \cdot o_c^{t-1} + (1 - f_c) \cdot o_c'. \end{aligned} \tag{5}$$

(K2.)[1] Except from semantics, relation knowledge also needs to be combined with contextual information. For relation in PG_t, we consider commonsense remains unchanged when solving a problem (e.g., "banana" always belongs to "fruit" in a problem), i.e., $ww_{i,j}^t = ww_{i,j}^{t-1}$. Therefore, we only adjust the word-operator relation $wo_{i,c}^t$ by:

$$\begin{aligned} wo_{i,c}' &= softmax(ReLU(\mathbf{W}_e \cdot [h_i^t, o_c^t, wo_{i,c}^{t-1}] + b_e)), \\ f_{i,c} &= \sigma(\mathbf{W}_{fe} \cdot [h_i^t, o_c^t] + b_{fe}), \\ wo_{i,c}^t &= f_{i,c} \cdot wo_{i,c}^{t-1} + (1 - f_{i,c}) \cdot wo_{i,c}'. \end{aligned} \tag{6}$$

(K3.) To utilize the information of words and operators for better generating the next reasoning goal, we propagate down from the top of PG_t by another HGCN:

$$\begin{aligned} AGG_i &= \left[\sum\nolimits_{c=1}^{C} wo_{i,c}^t \cdot o_c^t, \sum\nolimits_{j=1}^{n} ww_{i,j}^t \cdot h_j^t, \sum\nolimits_{j \in \mathcal{N}_d(i)} h_j^t\right], \\ h_i' &= ReLU(\mathbf{W}_u' \cdot AGG_i + b_u'), \\ q_a &= \sum\nolimits_{i=1}^{n} wg_i^t \cdot h_i', \\ f &= \sigma(\mathbf{W}_{fg} \cdot [q_a, q_t] + b_{fg}), \\ q' &= f \cdot q_t + (1 - f) \cdot ReLU(\mathbf{W}_g \cdot q_a + b_g), \end{aligned} \tag{7}$$

where $\mathbf{W}_u'$ and $\mathbf{W}_u$ share the same parameters that propagate information between words, and q_a aggregates the information from all words to the goal. q' represents an interim goal that needs to be further solved (e.g., "How many fruits do Jack and Amy have respectively") after accomplishing the current one q_t (e.g., "How many fruits do they have"), which will be used to generate the next reasoning goal q_{t+1} in the commutative module below.

Symbol Prediction: The obtained context-related semantics $\{h_i^t, i = 1, \ldots, n\}$, $\{o_c^t, c = 1, \ldots, C\}$ have integrated existing knowledge with contextual information, and thus can better achieve the current goal q_t and reason y_t. Consequently, we input them into a pointer-generator network [37] to generate y_t at time t. Specifically, since y_t can to be inferred from the external vocabulary (i.e., $V_O \cup V_C$) or derived from the problem itself (i.e., N_P), we first compute the probability of y_t belonging to the external vocabulary:

$$\mathbf{P}_{gen} = \sigma(\mathbf{W}_{gen} \cdot [q_t, c_t] + b_{gen}), \tag{8}$$

where c_t is a context vector fusing the word and clause level semantics [11]. Then, we calculate the distribution of y_t on the external vocabulary (denoted as $\mathbf{P}_g(y_t)$) and N_P (denoted as $\mathbf{P}_p(y_t)$) by

$$\mathbf{P}_g(y) = softmax(w_{gs}^{\mathrm{T}} \cdot ReLU(\mathbf{W}_{ga} \cdot [q_t, c_t, e_y] + b_{ga})), \tag{9}$$

$$\mathbf{P}_p(y) = softmax(w_{ps}^{\mathrm{T}} \cdot ReLU(\mathbf{W}_{pa} \cdot [q_t, c_t, h_y^t] + b_{pa})), \tag{10}$$

respectively. In (9), $e_y = o_c^t$ if y is operator c, otherwise e_y is a learnable embedding. In (10), h_y^t denotes the representation of symbol y obtained from (4). In summary, the probability $\mathbf{P}(y_t \mid y_1, \ldots, y_{t-1}, H, q_t)$ in (2) is:

$$\mathbf{P}(y_t) = \begin{cases} (1 - \mathbf{P}_{gen}) \cdot \mathbf{P}_p(y_t) & \text{if } y_t \in N_P, \\ \mathbf{P}_{gen} \cdot \mathbf{P}_g(y_t) & \text{if } y_t \in V_O \cup V_C. \end{cases} \tag{11}$$

Commutative Module: After reasoning symbol y_t, our CogSolver needs to generate the next goal q_{t+1} to support the next reasoning step at time $t+1$. Specifically, if y_t is an operator, the commutative module will first decompose the interim goal q' obtained by (7) into a left sub-goal q^l (i.e., q_{t+1}) and infer the left child tree t^l. Then, the right sub-goal q^r of q' will be generated using q' and t^l. The left and right sub-goal q^l, q^r are calculated by networks proposed in [6] and we summarize them briefly as $q^l = Decompose1(q')$ and $q^r = Decompose2(q', t^l)$.

To further apply the *mathematic rule knowledge* to improve the goal decomposition results, we notice that when the commutative law is satisfied, it is equivalent to generate the left sub-goal first (e.g., "3" in Fig. 3) and the right sub-goal first (e.g., "2"). Thus, our intuitive idea to utilize this knowledge is to ensure that CogSolver generates the same reasoning goals from left to right as from right to left when y_t satisfies the commutative law. In concrete implementation, we inversely generate an extra left sub-goal q_{inv}^l using the right child tree t_r. If y_t is $+$ or $\times$, q_{inv}^l should convey the same meaning as q^l. In this case, we define the commutative loss $Loss_C$ in (3) as their distance calculated by (12). For other cases (e.g., y_t is $-$ or $\div$), $Loss_C = 0$.

$$Loss_C = || \underbrace{Decompose2(q', t^r)}_{q_{inv}^l} - \underbrace{Decompose1(q')}_{q^l} ||. \tag{12}$$

3) Knowledge Update: Note that during the learning process, the knowledge in *BRAIN* may contain some redundancies. For example, after initialization in Section IV-B1, the edge strength $wo_{i,c}$ between word "the" and operator "+" in BG may be very strong, which is unreasonable and will hurt the reasoning performance. Thus, it is necessary to simulate in CogSolver how humans update their acquired knowledge after solving problems, while guaranteeing its rationality.

Specifically, *semantics knowledge* (i.e., $\{w_i, o_c\}$ of nodes in BG) is naturally updated by back-propagation when minimizing (3), and *mathematic rule knowledge* (i.e., commutative law) is fixed. Thus, here we focus on updating *relation knowledge* (i.e., $\{ww_{i,j}, wo_{i,c}\}$ of edges in BG).

As for words i, j, we imply their word-word relation $ww_{i,j}$ by the distance between semantic vectors as follows:

$$ww_{i,j} = \sigma(-||w_i - w_j|| + Mean_{dis}), \tag{13}$$

where $Mean_{dis}$ is the average distance between all word pairs and is used to broaden the distribution of $ww_{i,j}$. However, (13)

[1] Dependency edges represent the semantic structure of P and remain unchanged at different reasoning steps. Edges wg_i^t are recomputed by the hierarchical attention mechanism [11] at each step t.

may lead to the inclusion of superfluous and unexplainable knowledge because its output belongs to [0,1], which means there is a relationship between any pair of words. To address this issue, we further introduce a knowledge filter with threshold $\delta_1 > 0$ to eliminate relationships with weak strength:

$$ww_{i,j} = \begin{cases} ww_{i,j} & \text{if } ww_{i,j} > \delta_1, \\ 0 & \text{else.} \end{cases} \tag{14}$$

Please note that our knowledge filter does not introduce any additional neural networks. This non-parametric design allows us to avoid the need for extra model training and ensures that we are not constrained by the filter's generalization capabilities. As demonstrated by the results of Section VI-D1, this simple design effectively and robustly achieves the desired knowledge update performance.

For $wo_{i,c}$, we calculate the word-operator relation between each word and all operators using softmax function:

$$wo_{i,c} = softmax(-||w_i - o_c||). \tag{15}$$

Moreover, if a word does not associate with any operator (e.g., "apple" is not related to $\{+, \times, -, \div\}$), its weights with different operators could be thought almost equal (e.g., $wo_{i,c} \approx 0.25$). Thus, we also include another knowledge filter with threshold $\delta_2 > 0$ to filter out $wo_{i,c}$ by:

$$wo_{i,c} = \begin{cases} wo_{i,c} & \text{if } wo_{i,c} > \delta_2, \\ 0 & \text{else.} \end{cases} \tag{16}$$

Note that the sum of the relation knowledge between a word and all operators may not equal 1. For example, the updated $wo_{i,c}$ between "apple" and each operator may equal 0, meaning that "apple" is unrelated to any operator and thus will not directly contribute to predicting the probability of operators by $wo_{i,c}$, which is reasonable in reality.

In summary, before learning, the knowledge in *BRAIN* is initialized randomly and does not have rational meaning. With the *Store-Apply-Update* steps being iteratively carried out when solving problems (i.e., autonomously knowledge learning process), it is constantly improved and finally forms a long-term mathematical knowledge base. We will demonstrate and visualize it in Section VI-D. Meanwhile, as the knowledge becomes more and more accurate, the effect of *ARM* in problem answering will also improve.

It should be emphasized that conceptually, our *Store-Apply-Update* steps are not restricted to either the model training or testing phases. Mimicking humans, they iterate to achieve a continuous learning process. In practice, we executed the *Store-Apply-Update* steps on the training data and used the final model (with fixed knowledge in the *BRAIN* system) to test in experiments for fair comparison.

V. CogSolver+: Modeling Knowledge Recall Mechanism

CogSolver can effectively mimic the fundamental processes of humans learning and applying mathematical knowledge to solve MWP. However, during the *Store-Apply-Update* learning iterations, its *BRAIN-ARM* systems may inevitably forget the previously mastered knowledge. That is because the *BRAIN* system updates knowledge directly based on the problems solved within the *apply* step at current iteration (denoted as $\mathcal{T}$) by (14) and (16), which may overwrite the experience accumulated from previous iterations $\{1, 2, \ldots, \mathcal{T}-1\}$ and results in unsatisfying knowledge forgetting. For example, after solving the problem "Jack has 3..." in Fig. 4, it may lost the grasp of the relationship (i.e., weaken $wo_{i,c}$ by (16)) between "orange" and "fruit" due to the updated semantics w_i of "fruit" (validated in Section VI-D1). Such a phenomenon is similar to the process of humans naturally forgetting what has been learned as claimed in psychological research (e.g., Ebbinghaus curve [58]), which attracts great attention as a catastrophic forgetting problem when continually learning various types of knowledge [28], [59], [60]. In order to address this crucial issue, we further draw insight from another prominent psychological theory: complementary learning system (CLS) theory [24], [25], which points out that there exists essential memory replaying underling human cognition to help retain the accumulated knowledge. Take the above example again for illustration, after solving the problem "Jack has 3...", we may recall a similar problem "Mary paid \$30 for berries and \$20 for oranges. How much money did she spend on fruits?" in our mind, because it also examines the knowledge relevant to "fruit". Then, by applying the weakened relationship between "orange" and "fruit" to solve it, we could realize that such an *updated* result is unreasonable and induce targeted consolidation (i.e., reinforce $wo_{i,c}$) to avoid the degradation. To this end, in this section, we aim to simulate human memory replay and introduce an indispensable knowledge recall mechanism into our CogSolver, extending it to CogSolver+ to facilitate the consolidation of high-quality long-term knowledge.

More specifically, as the CLS theory reveals, human brain figures out the forgetting problem through the synergistic interactions between hippocampus and neocortex [26], where the hippocampus replays experiences (i.e., previously solved problems in our scenario) to the neocortex and the neocortex consolidates the long-term knowledge based on it [26], [27], [28]. Along this line, we carry out the knowledge *recall* mechanism in a "Replay-Consolidate" manner as depicted in Fig. 4. The Replay process (Section V-A) is responsible for replaying problems relevant to the current learning iteration, and the Consolidate process (Section V-B) reinforces or weakens the updated knowledge by assessing whether it contributes to solving the replayed problems.

A. Problem Replay

We first introduce the problem replay process in Fig. 4, i.e., retrieve a set of problems D_{re} from problems that have been solved in *Store-Apply-Update* iterations before $\mathcal{T}$ (denoted as D, $D_{re} \subseteq D$). Pedagogically and psychologically, we consider three key factors into constructing D_{re}: knowledge relevance, problem relevance, and temporal relevance [20], [58], [61], [62]. Knowledge relevance reflects the similarity in knowledge between the replayed problems and problems solved in the current iteration $\mathcal{T}$ (specifically, within the *apply* step). For instance, it is easy to bring back a problem like "Mary paid..." into D_{re} after

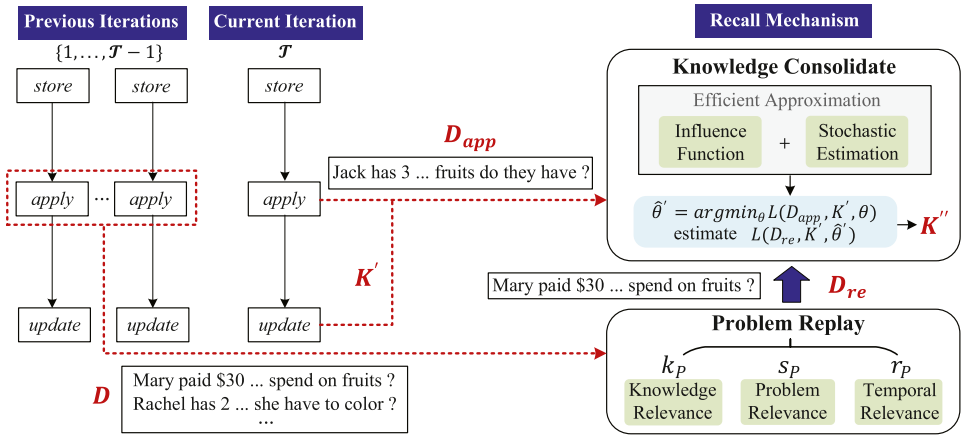


Fig. 4. Knowledge recall mechanism, which operates in a Problem Replay-Knowledge Consolidate manner.

solving the problem "Jack has..." in Fig. 4, because they both examine the knowledge related to "fruit". Problem relevance measures the similarity of problem text. For example, in Fig. 4, another problem "Rachel has 2 coloring books. One had 24 pictures and the other had 39. How many pictures does she have to color?" is also likely to be replayed, because both of it and "Jack has..." aim to calculate the total number of items and thus is similar in semantics. As for temporal relevance, researches have converged that the difficulty of remembering increases with the passage of time [58]. Therefore, if a problem appears in an early learning iteration, it has a lower probability of being traced back into D_{re}.

We model these three factors separately and then consider them jointly to estimate a probability $\mathbf{P}_{re}(P)$ of retrieving each problem $P \in D$ into D_{re} as follows. Specifically, for knowledge relevance, since our knowledge is mostly defined with respect to words (e.g., $\{w_i, ww_{i,j}, wo_{i,c}\}$), it is rational to leverage the words in the problem sentence to measure a correlation, e.g., two problems containing the word "fruit" are similar in terms of examined knowledge. In specific, for two problems P and Q, we represent them as the TF-IDF vectors $v_P, v_Q \in \mathbb{R}^N$ (N is the number of words in the field of MWP defined in Section IV-B1) and estimate their knowledge relevance by cosine similarity:

$$k_{P,Q} = \cos(v_P, v_Q). \tag{17}$$

To calculate the overall knowledge relevance of P, we average $k_{P,Q}$ for all problems being solved at current knowledge *apply* step (denoted as D_{app}):

$$k_P = \frac{1}{|D_{app}|} \sum_{Q \in D_{app}} k_{P,Q}. \tag{18}$$

For problem relevance, we input the problem P into the encoder f_θ in (1) and take the initial reasoning goal q_1 as its representation. Compared with knowledge relevance that uses TF-IDF, it excavates semantics on the whole and focuses more on what the problem aims to solve. Thus, they capture different level of memory aspects and are both necessary. Formally, we measure problem relevance s_P by:

$$s_P = \frac{1}{|D_{app}|} \sum_{Q \in D_{app}} \cos(q_P, q_Q). \tag{19}$$

For temporal relevance, we record the learning iteration in which problem P was solved and calculate its difference from the current iteration. We denote the result as $r_P \in \{1, 2, \ldots, \mathcal{T} - 1\}$. Please note that with the increase of r_P, problem P is more difficult to recall as it was solved in a more distant iteration, while the larger k_P and s_P, the more relevant it is to the current iteration. Therefore, in Fig. 4, we combine the three factors k_P, s_P, and r_P in the following way and sample U problems from D into the memory set D_{re} from the below distribution:

$$\mathbf{P}_{re}(P) = softmax\left(\frac{k_P \cdot s_P}{\sqrt{r_P}}\right), P \in D. \tag{20}$$

B. Knowledge Consolidation

Based on the retrieved memory D_{re}, now we simulate how humans consolidate the updated knowledge, e.g., reinforce/weaken the reasonable/irrational knowledge. Without loss of generality, we first define some basic notations. We use K, K' to represent the knowledge before and after current knowledge *update* step respectively. $\mu \triangleq K' - K$ is their difference, i.e., the increment during the *update* step. Given K, K', we denote $\hat{\theta} \triangleq \operatorname{argmin}_\theta L(D_{app}, K, \theta), \hat{\theta}' \triangleq \operatorname{argmin}_\theta L(D_{app}, K', \theta)$ as the optimal model parameters of CogSolver+ trained on K and K', respectively, where L is short for the loss function $Loss$ in (3). Please note that $\hat{\theta}$ is naturally obtained when minimizing (3) in the knowledge *apply* step, while $\hat{\theta}'$ does not.

To determine whether to reinforce or weaken the updated knowledge K', our basic idea is applying K' to solve the replayed problems D_{re} and evaluating the expression reasoning performance, i.e., $L(D_{re}, K', \hat{\theta}')$. Please note that here the parameter is $\hat{\theta}'$ rather than $\hat{\theta}$, because $\hat{\theta}'$ implies the best way to apply knowledge K' learned from minimizing L. In general, if $L(D_{re}, K', \hat{\theta}')$ is smaller enough, the updated K' is simultaneously beneficial for solving the replayed problems and thus

should be reinforced. On the contrary, if $L(D_{re}, K', \hat{\theta}')$ is too large, the updated knowledge K' can not generalize well and should be weakened.

However, it is inefficient to reestimate $\hat{\theta}'$ because it relies on an additional optimization process, which has time complexity $O(MT\Theta)$ where $M = |D_{app}|$, T is a predefined number of epoches of an optimization algorithm (e.g., Adam [63]), and $\Theta = |\theta|$ is the number of parameters (i.e., $\hat{\theta}, \hat{\theta}' \in \mathbb{R}^{\Theta}$). In particular, to make the optimization converge, we find that there is a requirement for T (proof can be found in Section 1 of Supplementary Material):

Theorem 1: Assume $\frac{\partial L}{\partial \theta}$ is locally Lipschitz continuous with the Lipschitz constant upper bounded by $W > 0$. Then, to find the optimal $\hat{\theta}'$, we have $T > O(\sqrt{\frac{\|K'-K\|^2+\|\widehat{\theta}'-\widehat{\theta}\|^2}{W \cdot \|\widehat{\theta}'-\widehat{\theta}\|^2}})$. Based on Theorem 1, it necessitates at least $O(M\sqrt{\mathcal{K}^2+\Theta^2})$ to find the precise $\hat{\theta}'$, where $\mathcal{K}$ is the number of knowledge. Such a complexity of M, $\mathcal{K}$ and Θ is infeasible for sheer volumes of datasets, knowledge, and model scale in practice. Consequently, in the following, we go back to our original goal, i.e., estimate $L(D_{re}, K', \hat{\theta}')$, and turn to find an efficient approximation of it. Specifically, we begin with adopting a first order Taylor approximation of $L(D_{re}, K', \hat{\theta}')$ with respect to $\mu = 0$ as

$$L\left(D_{re}, K', \hat{\theta}'\right) \approx L\left(D_{re}, K, \hat{\theta}\right) + \mu \cdot I\left(K', D_{re}\right)$$

$$I(K', D_{re}) \triangleq \frac{dL\left(D_{re}, K', \widehat{\theta}'\right)}{d\mu}|_{\mu=0} = \nabla_K L\left(D_{re}, K, \hat{\theta}\right) + \nabla_\theta L\left(D_{re}, K, \hat{\theta}\right)^T \cdot \frac{d\hat{\theta}'}{d\mu}\bigg|_{\mu=0}. \tag{21}$$

In (21), it is easy to calculate the gradients $\nabla_K L(D_{re}, K, \hat{\theta})$ and $\nabla_\theta L(D_{re}, K, \hat{\theta})$ since we have obtained K and $\hat{\theta}$, so now our goal turns to estimate $\frac{d\hat{\theta}'}{d\mu}|_{\mu=0}$. Specifically, we find that this term is actually estimating the influence of knowledge perturbation on optimizing CogSolver+'s parameters. Therefore, inspired by [27], [64], we introduce an auxiliary $\hat{\theta}'' \triangleq \operatorname{argmin}_\theta L(D_{app}, K, \theta) + \varepsilon L(D_{app}, K', \theta) - \varepsilon L(D_{app}, K, \theta)$, which serves as an intermediary to help investigate the influence of upweighting the updated knowledge K' during training (with a small term ϵ) on parameters θ. Given $\hat{\theta}''$, we can derive a first-order influence function analogous to [65] as:

$$\frac{d\widehat{\theta}''}{d\varepsilon}|_{\varepsilon=0} \approx -H_{\hat{\theta}}^{-1}\left[\nabla_\theta L\left(D_{app}, K', \hat{\theta}\right) - \nabla_\theta L\left(D_{app}, K, \hat{\theta}\right)\right], \tag{22}$$

where $H_{\hat{\theta}} \triangleq \nabla_\theta^2 L(D_{app}, K, \hat{\theta})$ is the Hessian matrix and is positive definite because $\hat{\theta}$ is the optimal parameter with respect to knowledge K on problem D_{app}.

For the left-hand side of (22), we notice that $\hat{\theta}' = \hat{\theta}''|_{\varepsilon=1}, \hat{\theta} = \hat{\theta}''|_{\varepsilon=0}$. For the right-hand side, we adopt another Taylor approximation $\nabla_\theta L(D_{app}, K', \hat{\theta}) - \nabla_\theta L(D_{app}, K, \hat{\theta}) \approx \nabla_K \nabla_\theta L(D_{app}, K, \hat{\theta}) \cdot \mu$ since $\|\mu\| \to 0$. On this basis, we rewrite (22) into

$$\hat{\theta}' - \hat{\theta} \approx -H_{\hat{\theta}}^{-1} \nabla_K \nabla_\theta L\left(D_{app}, K, \hat{\theta}\right) \cdot \mu. \tag{23}$$

Applying (23) into (21), we finally obtain:

$$I(K', D_{re}) = \nabla_K L\left(D_{re}, K, \hat{\theta}\right) - \nabla_\theta L\left(D_{re}, K, \hat{\theta}\right)^T \cdot H_{\hat{\theta}}^{-1} \cdot \nabla_K \nabla_\theta L\left(D_{app}, K, \hat{\theta}\right) \tag{24}$$

In (24), it requires $O(\Theta^3 + \Theta^2)$ operations to invert $H_{\hat{\theta}}$ and calculate $\nabla_\theta L(D_{re}, K, \hat{\theta})^T \cdot H_{\hat{\theta}}^{-1}$. Such complexity is prohibitively expensive, so we adopt a stochastic estimation [64] to compute it in $O(r\gamma\Theta)$ time, which recursively samples problems from D_{re} to achieve an asymptotic unbiased estimation. Here, r, γ are hyperparameters controlling the recursions to reduce the variance, which have great influence on the effect of knowledge consolidation (will be validated in Section VI-E). The pseudo-code of estimating $I(K', D_{re})$ is presented in Algorithm 1.

Algorithm 1: Estimation of $I(K', D_{re})$.

Input: $K,\ K',\ \hat{\theta},\ D_{re},\ D_{app}$
Output: $I(K', D_{re})$
1: **for** $i = 1, 2, \ldots, \gamma$ **do**
2: Initialize $\tilde{H}_0 = \nabla_\theta L(D_{re}, K, \hat{\theta})^T$;
3: **for** $j = 1, 2, \ldots, r$ **do**
4: Uniformly sample a data z_j from D_{app};
5: $\tilde{H}_j = \nabla_\theta L(D_{re}, K, \hat{\theta})^T + \tilde{H}_{j-1} \cdot (I - \nabla_\theta^2 L(z_j, K, \hat{\theta}))$;
6: $\tilde{H}^i \leftarrow \tilde{H}_r$;
7: Calculate $I(K', D_{re})$ by (24), where $\nabla_\theta L(D_{re}, K, \hat{\theta})^T \cdot H_{\hat{\theta}}^{-1} = \frac{1}{\gamma}\Sigma_{i=1}^{\gamma}\tilde{H}^i$.

In summary, based on $I(K', D_{re})$, we have derived an efficient estimation of $L(D_{re}, K', \hat{\theta}')$ through (21) and (24), which only relies on the current model parameter $\hat{\theta}$ and does not require any retraining based on the updated knowledge K'. Since $\nabla_\theta L(D_{re}, K, \hat{\theta})^T \cdot H_{\hat{\theta}}^{-1}$ is the same for all knowledge and only needs to be calculated once, the overall time complexity decreases from $O(M\sqrt{\mathcal{K}^2+\Theta^2})$ to $O(r\gamma\Theta)$, where $r, \gamma \ll \mathcal{K}, M, \Theta$.

In specific, as $\mu \cdot I(K', D_{re})$ decreases, $L(D_{re}, K', \hat{\theta}')$ becomes smaller, indicating that the updated knowledge K' can generalize well on the replayed memory D_{re} and thus should be reinforced [27]. On the contrary, a large $\mu \cdot I(K', D_{re})$ brings a higher expression generation (i.e., MWP solving) loss $L(D_{re}, K'\hat{\theta}')$, reflecting the poor robustness of the updated knowledge that should be weakened as a result. On this basis, to balance the result of knowledge *update* step with knowledge *recall* mechanism, we design a momentum strategy to adjust K' by:

$$K'' = \begin{cases} K' + \alpha \cdot (K' - K) & \text{if} \quad \mu \cdot I\left(K', D_{re}\right) \le -\delta_3, \\ K' - \alpha \cdot (K' - K) & \text{if} \quad \mu \cdot I\left(K', D_{re}\right) \ge \delta_3, \\ K' & \text{else}, \end{cases} \tag{25}$$

TABLE I
THE STATISTICS OF DATASETS

Dataset	Math23K	MAWPS	GSM8K
Number of problems	23,162	2,373	8,792
Number of operators	5	4	4
Average problem length	28.02	30.08	45.88
Average expression length	5.55	4.20	8.21

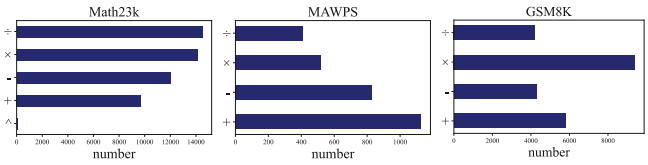


Fig. 5. The distribution of operators.

where $\delta_3 > 0$ is a positive threshold ensuring when $\mu \cdot I(K', D_{re}) \leq -\delta_3, L(D_{re}, K', \hat{\theta}') < L(D_{re}, K, \hat{\theta})$ in (21), and $\alpha \in [0, 1)$ is a momentum coefficient.

After completing (25), K'' is further stored in the *BRAIN* system and the *Store-Apply-Update* steps continue, with the knowledge *Recall* mechanism running after each learning iteration. In experiments, similar to CogSolver, our knowledge *recall* mechanism in CogSolver+ (including problem replay and knowledge consolidation) is also only conducted on the training set for fair comparison.

VI. EXPERIMENTS

In this section, we conduct extensive experiments to validate our proposed CogSolver and CogSolver+. Specifically, we first describe the datasets and introduce the experimental setups (Sections VI-A and VI-B). Then, we demonstrate the effectiveness of our models from various aspects: (1) the performance on answer reasoning compared with the baselines (Section VI-C); (2) the interpretable knowledge learning results (Section VI-D); (3) the effectiveness and efficiency of the knowledge recall mechanism in CogSolver+ (Section VI-E); (4) the explainable reasoning processes (Section VI-F).

A. Dataset Description

We conducted experiments on three real-world datasets: **Math23 K**, **MAWPS**, and **GSM8K**. Math23K [4] is a well-known dataset that contains 23,162 Chinese math word problems for elementary school students. Wu et al. [38] have also published an external commonsense knowledge graph for it. MAWPS [66] is an English dataset that contains 2,373 problems with one unknown variable. GSM8K [67] is a popular benchmark, containing 8,792 high-quality, diverse, and more difficult problems. However, the original GSM8K dataset does not provide the mathematical expressions required as training labels in MWP setting for both the baselines and our models. To facilitate the experiments, we first use GPT4 to convert the original solutions expressed in natural language into prefix expressions and manually revise them to ensure that they obtain the correct numeric answers. We released all the data and code at https://github.com/Ljyustc/CogSolver. Since Math23 K and GSM8K have provided a train/test partition when published, we follow their original setup for evaluation. For MAWPS, we conduct 5-fold cross-validation. Table I summarizes the basic statistics of the three datasets, and Fig. 5 shows the distribution of operators occurring in target expressions. We can find that except for $\wedge$ (power), $\{+, \times, -, \div\}$ are distributed more evenly in the Math23K, while $+$ and $\times$ accounts for nearly 40% in MAWPS and GSM8K, respectively. It will bring the challenge of learning knowledge related to these operators.

B. Experimental Setup

We implement our CogSolver and CogSolver+ in both non-pre-train (**CogSolver**, **CogSolver+**) and pre-train settings (**CogSolver(B)**, **CogSolver+(B)**). Implementation details are presented in Section 2 of Supplementary Material.

1) Evaluation Metrics: We use answer accuracy (Ans ACC) and equation accuracy (Equ ACC) as the evaluation metrics. The answer accuracy refers to whether the calculation result of the predicted expression equals the ground truth, while the equation accuracy requires that the predicted expression is exactly the same as the annotated one in the dataset.

2) Baseline Approaches: To demonstrate the effectiveness of our models, we select 11 representative and state-of-the-art methods as baselines:

- **DNS** [4]: uses a vanilla seq2seq model to translate a problem into an equation directly.
- **Math-EN** [68]: utilizes equation normalization to address duplicated equations.
- **T-RNN** [69]: first applies a seq2seq model to predict a tree-structure template and then designs a recursive neural network to infer the unknown operator nodes.
- **GROUP-ATT** [70]: introduces a group attention mechanism to improve the capability of extracting multiple types of features from problems.
- **GTS** [6]: adopts a heuristic goal-driven mechanism and proposes a tree-structured neural network to generate expression trees in a seq2tree manner.
- **Graph2Tree** [7]: designs a graph-based encoder to enrich the understanding of quantities by capturing the relationships and order information among them.
- **KA-S2T** [38]: incorporates manually constructed external commonsense knowledge bases to obtain better problem representations.
- **HMS** [11]: simulates human reading habits by leveraging a hierarchical word-clause-problem encoder to better exploit the problem semantics.
- **BERT-Tree** [10]: employs BERT as the encoder and GTS as the decoder and uses contrastive learning to perceive the divergent expression prototypes.
- **SUMC-Solver** [71]: designs a M-tree coding a seq2code method to unify equivalent expressions.
- **MWP-BERT** [40]: introduces several numeracy grounded pre-training objectives to capture numerical properties in masked language models.

TABLE II
ACCURACY PERFORMANCE ($*/\dagger : P \leq 0.01/0.001$)

	Model	Math23K		MAWPS		GSM8K	
		Ans ACC	Equ ACC	Ans ACC	Equ ACC	Ans ACC	Equ ACC
non-pre-train	DNS	0.581	0.568	0.595	0.582	0.077	0.065
	Math-EN	0.667	0.656	0.692	0.690	0.150	0.095
	T-RNN	0.669	0.650	0.668	0.656	0.108	0.080
	GROUP-ATT	0.695	0.556	0.761	0.746	0.100	0.068
	GTS	0.743	0.633	0.810	0.804	0.198	0.129
	Graph2Tree	0.764	0.647	0.820	0.809	0.143	0.084
	KA-S2T	0.763	0.652	/[2]	/[2]	/[2]	/[2]
	HMS	0.755	0.643	0.804	0.731	0.131	0.081
	CogSolver	**0.773***	**0.654**	**0.829***	0.768	**0.202**	**0.149**†
	CogSolver+	**0.780**†	**0.658**	**0.838**†	**0.816***	**0.208***	**0.151**†
pre-train	BERT-Tree	0.832	0.718	0.872	0.859	0.202	0.131
	SUMC-Solver	0.825	/[3]	0.820	/[3]	/[3]	/[3]
	MWP-BERT	0.836	0.713	0.829	0.820	0.183	0.125
	CogSolver(B)	**0.836**	0.711	**0.875**	**0.868***	**0.216**†	**0.156**†
	CogSolver+(B)	**0.843***	0.715	**0.883***	**0.870***	**0.227**†	**0.173**†

C. Performance on Answer Reasoning

1) Overall Accuracy: Table II presents the results of all models[4], and we find several key observations. First, our CogSolver and CogSolver+ surpass all baselines in both pre-train and non-pre-train scenarios, with improvements over the SOTA Graph2Tree, BERT-Tree, and MWP-BERT deemed statistically significant with $p \leq 0.01/0.001$ on both datasets according to paired t-test (indicated by "$*/\dagger$ "). These results clearly validate that they can benefit from applying the learned knowledge to obtain more accurate reasoning results, thereby validating the effectiveness of our proposed *BRAIN-ARM* systems and *Store-Apply-Update* steps. Second, CogSolver+ achieves better performance compared with CogSolver, which directly emphasizes the necessity and superiority of incorporating our knowledge *Recall* mechanism into the learning process for accumulating long-term, rational knowledge. Third, the comparison between models employing the same encoder (i.e., CogSolver+/CogSolver+(B) vs HMS/BERT-Tree) underscore the advantages of the knowledge-aware module and commutative module in augmenting the reasoning capabilities of *ARM*'s decoder, because they attain better results based on the same problem understanding. Especially, our advantage is more significant on the GSM8K dataset (e.g., CogSolver+(B) shows an accuracy improvement of 1.1% on MAWPS and 2.5% on GSM8K compared to BERT-Tree). This demonstrates that our models maintain strong robustness even when dealing with more challenging problems. Fourth, a notable observation is that CogSolver+ and CogSolver exhibit superior performance compared to KA-S2T which leverages manually constructed external commonsense knowledge bases. This is probably because our models can acquire more relevant and appropriate knowledge for MWP, such as gaining word-operator knowledge and consolidating knowledge based on MWP's reasoning objective. Besides, it reflects our models' capacity to learn knowledge from scratch and autonomously form a reliable knowledge base, which has better flexibility. Last but not least, CogSolver+ and CogSolver achieve a higher accuracy on MAWPS than Math23K, followed by GSM8K. The reason could be that "+" takes a larger proportion on MAWPS as depicted in Fig. 5. Consequently, it becomes more straightforward and easier for the models to learn and apply the relational knowledge between "+" and words for reasoning, implying that training data has a significant impact on knowledge learning results, which is further witnessed in Section VI-D2. As for GSM8K, aside from its inherent difficulty, we found that the diverse expression structures and the tendency for step omissions in the GPT4-generated expressions posed challenges for model training. However, under these conditions, our models still outperformed others, demonstrating their effectiveness and robustness.

2) Ablation Study: To provide an in-depth analysis of our methods, we perform an ablation study of CogSolver and CogSolver+ in Tables III and IV, respectively. Specifically, for knowledge storage, "w/o Rel" omits the *relation knowledge* $\{ww_{i,j}, wo_{i,c}\}$ in *BRAIN*, thereby eliminating the problem graph and knowledge-aware module, while "w/o Math-Rule" disregards $Loss_C$ in the commutative module (3) during training. For knowledge application, "w/o Know-aware Sem" and "w/o Know-aware Rel" do not calculate context-related semantics by (4), (5) and relationships by (6) in the knowledge-aware module, respectively. For knowledge update, "w/o update Rel" fixes the *relation knowledge* after initialization. Please note that in CogSolver+, we only recall the updated relation knowledge as stated in implementation details, hence there are no "w/o Rel" and "w/o update Rel" for it. For knowledge recall, "w/o Know-Relevance", "w/o Prob-Relevance", and "w/o Temp-Relevance" indicate eliminating the three factors k_P, s_P, r_P in problem replay process (20), respectively.

We conclude the results as follows. First, our proposed knowledge learning steps and mechanism are necessary for correctly reasoning the answers, because the accuracies of CogSolver and CogSolver+ degrade when any component is omitted. Second, compared with "w/o Math-Rule", "w/o Rel" diminishes the results more greatly, indicating that *relation knowledge* is more effective than commutative law in facilitating reasoning, which further highlights the importance of updating it during the learning process. Third, the performance of CogSolver suffers the most considerable deterioration when the knowledge is not updated (i.e., "w/o update Rel"). It indicates that knowledge update plays the most crucial role in CogSolver, possibly because it can eliminate wrong knowledge that harms the reasoning process. Forth, context-related semantics and relation are almost equally important as there is no significant difference between "w/o Know-aware Sem" and "w/o Know-aware Rel". Last, the temporal relevance has the greatest influence on knowledge recall, which suggests that earlier knowledge is indeed more likely to be forgotten (and thus need to be recalled more frequently), confirming the necessity and effectiveness of our proposed recall mechanism.

[2]KA-S2T [38] does not provide knowledge base or conduct experiment on MAWPS and GSM8K.

[3]SUMC-Solver [71] generates M-codes, rather than equations for getting the answers, and thus does not have Equ ACC. Besides, it does not provide M-Trees for GSM8K, so there are no corresponding results.

[4]To get the results of Equ ACC, we rerun the original codes of baselines with Pytorch version 1.8.1. Besides, we implement BERT-Tree [10], and MWP-BERT [40] on MAWPS (not reported in the original papers) by adapting MAWPS to the data format set in the source codes.

TABLE III
ABLATION STUDY OF COGSOLVER IN TERMS OF ANSWER ACCURACY

Model			Math23K		MAWPS		GSM8K	
			non-pre-train	pre-train	non-pre-train	pre-train	non-pre-train	pre-train
CogSolver			**0.773**	**0.836**	**0.829**	**0.875**	**0.202**	**0.216**
Store	w/o Rel		0.754	0.827	0.809	0.872	0.186	0.205
	w/o Math-Rule		0.766	0.834	0.820	0.873	0.194	0.211
Apply	w/o Know-aware	Sem	0.758	0.830	0.820	0.869	0.189	0.207
		Rel	0.761	0.830	0.818	0.872	0.193	0.205
Update	w/o update Rel		0.741	0.822	0.796	0.862	0.178	0.192

TABLE IV
ABLATION STUDY OF COGSOLVER+ IN TERMS OF ANSWER ACCURACY

Model			Math23K		MAWPS		GSM8K	
			non-pre-train	pre-train	non-pre-train	pre-train	non-pre-train	pre-train
CogSolver+			**0.780**	**0.843**	**0.838**	**0.883**	**0.208**	**0.227**
Store	w/o Math-Rule		0.770	0.825	0.831	0.878	0.202	0.219
Apply	w/o Knowl-aware	Sem	0.768	0.830	0.827	0.871	0.194	0.204
		Rel	0.765	0.837	0.828	0.873	0.184	0.199
Recall	w/o Know-Relevance		0.770	0.829	0.833	0.867	0.198	0.202
	w/o Prob-Relevance		0.771	0.837	0.832	0.868	0.201	0.217
	w/o Temp-Relevance		0.763	0.830	0.829	0.861	0.192	0.200

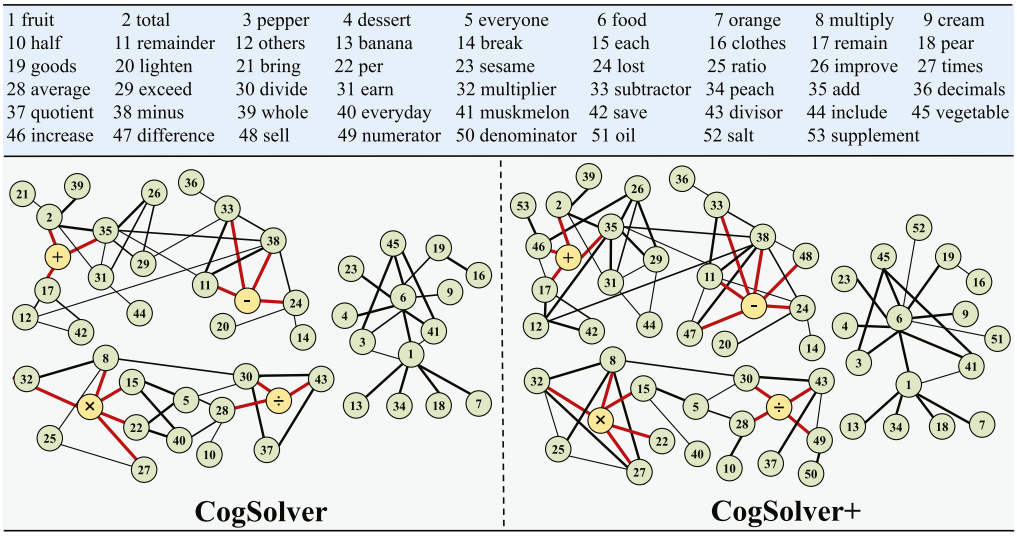


Fig. 6. Part of BG in *BRAIN* of CogSolver and CogSolver+ after autonomously learning on Math23K. The green and yellow nodes represent words and operators respectively. The black and red line represent word-word relation $ww_{i,j}$ and word-operator relation $wo_{i,c}$ respectively. The width of line reflects the relation strength, i.e., the thicker the line, the greater the strength of the relation.

D. Analysis on Knowledge Learning

The key capability of our CogSolver and CogSolver+ is to autonomously learn mathematical knowledge from solving MWP through our proposed *Store-Apply-Update* steps and knowledge *Recall* mechanism. To verify it, we visualize part of the acquired BG in *BRAIN* (Section IV-B1) after learning from scratch in Fig. 6.[5] In general, we observe that both CogSolver and CogSolver+ have formed reasonable knowledge bases that contain word-word and word-operator relation. For instance, on Math23K, CogSolver learns that "+" is related to word "add", "add" is related to "exceed", and "+" is not related to "multiply", while CogSolver+ additionally learns that "+" is related to "increase". The line width signifies the relation strength (e.g., the relationship between "add-total" is stronger than "add-earn" for CogSolver). It is noteworthy that the knowledge acquired by different models exhibits commonalities and characteristics. For example, both CogSolver and CogSolver+ learn the strong relationship between "total" and "+", while CogSolver+ considers "increase" as more relevant to "+ " than "remain". This observation reveals the potential to merge learning results of different models for constructing a more comprehensive and robust knowledge base. In the following, we take Math23K and MAWPS datasets as examples to analyze the word-word and word-operator relation knowledge in *BRAIN* in detail to validate

[5]Results on MAWPS and GSM8K are similar to Math23K, so we omit them to avoid redundancy.

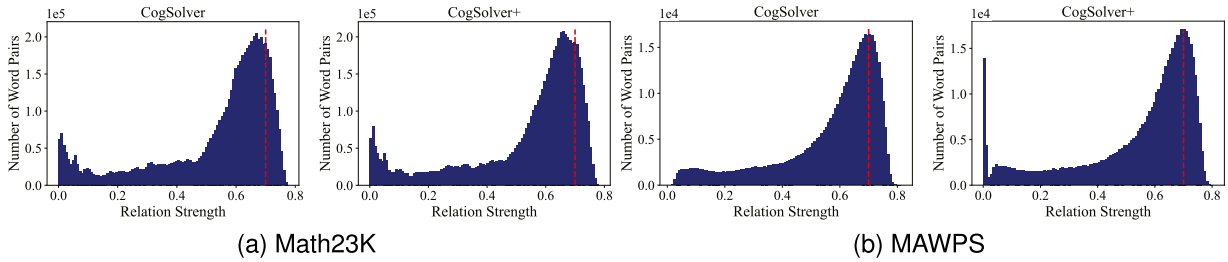


Fig. 7. Distribution of word-word relation $ww_{i,j}$. The red dash line represents the threshold $\delta_1 = 0.7$ in (14).

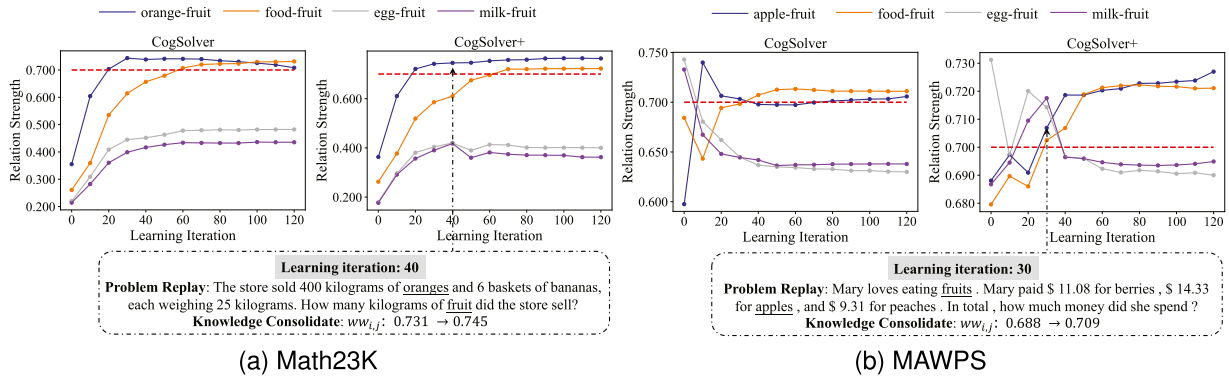


Fig. 8. Word-word relation $ww_{i,j}$ (examples). The red dash line represents the threshold $\delta_1 = 0.7$ in (14).

the effectiveness of CogSolver and CogSolver+ on knowledge learning.

1) Word-Word Relation: We compute the learned relation strength $ww_{i,j}$ between any two words by (13) and visualize the distribution for all word pairs in Fig. 7. We can see on both datasets, most of the relationships acquired by CogSolver and CogSolver+ exhibit a strength of approximately $0.6 \sim 0.7$. With threshold $\delta_1 = 0.7$ in (14), about 15% and 20% of word pairs (right of the red dash line) are retained (have commonsense knowledge) on Math23K and MAWPS, respectively. To assess their quality quantitatively, we further compare them with manually constructed knowledge sources, utilizing the knowledge base specifically designed for Math23K (published by KA-S2T [38]) and ConceptNet [72] for MAWPS. We find that on Math23K, approximately 77.2% and 80.6% of true commonsense knowledge has been automatically acquired by CogSolver and CogSolver+, while on MAWPS, the respective percentages are 74.7% and 75.2%. Therefore, our models demonstrate the capacity to accurately learn word-word relation knowledge while maintaining superior flexibility and generality across diverse datasets.

Furthermore, we take multiple word pairs as examples and illustrate the evolution of their relation strength during the learning process in Fig. 8. As we know, there exist real relationships between "orange-fruit", "food-fruit", and "apple-fruit", while there are invalid relationships between "egg-fruit" and "milk-fruit". For the real relationships, we observe they gradually strengthen and eventually exceed the threshold $\delta_1 = 0.7$ (the red dash line). On the contrary, for the invalid relationships, despite their divergent initialization on Math23K (≈ 0.20) and MAWPS (e.g., ≈ 0.73 for "egg-fruit"), their strength remains lower than both the real relationships and the threshold after learning, thus not be retained in *BRAIN*. These results show that our knowledge filters in both CogSolver and CogSolver+ are capable and robust enough to distinguish between different relationships and carefully filter out those invalid.

In particular, we note that for CogSolver, the relational strength between "orange-fruit" on Math23 K has a decreasing trend after the 40-th learning iteration (left part of Fig. 8(a)). Similarly, the strength between "apple-fruit" declines on MAWPS at the 10-th iteration (left part of Fig. 8(b)). This observation confirms the existence of the knowledge forgetting phenomenon we discussed in Section V. In contrast, CogSolver+ avoids such unwarranted degradation by replaying the memory of problems and adjusting the updated knowledge $ww_{i,j}$. For example, as shown in the right part of Fig. 8(b), based on the replayed problem "Mary loves eating fruits... for apples... she spend ?" in the 30-th iteration, it strengthens the updated $ww_{i,j}$ between "apple-fruit" from 0.688 to 0.709 on MAWPS, which not only addresses the forgetting of the knowledge strength but also enhances the subsequent knowledge application during the learning process, enabling the continual reinforcement of it. These results provide direct evidence that our proposed knowledge recall mechanism is essential and can effectively consolidate reasonable long-term knowledge.

2) Word-Operator Relation: We count the number of words related to each operator after learning and select the top 3 with the highest relation strength $wo_{i,c}$ by (16) for CogSolver+ in

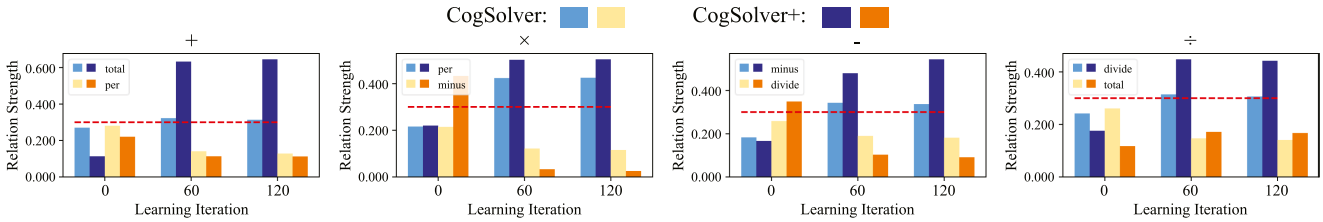


Fig. 9. Word-operator relation $wo_{i,c}$ (examples) on Math23K. The red dash line represents the threshold $\delta_2 = 0.3$ in (16).

TABLE V
COGSOLVER+: NUMBER OF WORDS AND TOP 3 WITH THE GREATEST RELATION STRENGTH FOR EACH OPERATOR

	Math23k		MAWPS	
	Num	top 3 words	Num	top 3 words
+	7	total,increase,add	328	adds,earns,total
−	61	difference,minus,sell	15	gave,sold,lost
×	7	multiply,ratio,each	19	each,pieces,times
÷	5	divide,average,numerator	13	each,half,split

TABLE VI
PERFORMANCE COMPARISON BETWEEN COGSOLVER+ AND COGSOLVER+(O) ON MATH23 K

	CogSolver+	CogSolver+(O)			
		$T = 1$	$T = 10$	$T = 50$	$T = 100$
Ans ACC	0.780	0.750	0.752	0.767	0.777
Time(s)	412.88	757.90	2602.89	2549.83	3221.54

Table V (results of CogSolver can be found in Section 3 of Supplementary Material). For instance, CogSolver+ learns that 61 words are related to "− " on Math23K, with the top 3 being "difference", "minus", and "sell". We can see that these words do imply the meanings of the relevant operators, indicating that its acquired relationships are rational. Besides, on MAWPS, only 15, 19, and 13 words remain related to "−","×", and "÷" respectively for CogSolver+ (similar for CogSolver), compared to 328 for +. Combining it with the finding in Section VI-C1, we conclude that the data volume has an important impact on the learning results.

To gain a deeper understanding of CogSolver's and CogSolver+'s competence in acquiring the knowledge, we select examples for $\{+, \times, -, \div\}$ on Math23K, as the operator distribution (excluding $\wedge$) on it is relatively uniform (Fig. 5). From Fig. 9, the relationships between "total"-"+", "per"-"×", "minus"-"−", and "divided"-"÷" are strengthened and stored in *BRAIN* after learning since they exceed the threshold $\delta_2 = 0.3$ for both models. In comparison, relationships between "per" and "+", 'minus" and "×", and so on gradually weaken over time. It suggests that CogSolver and CogSolver+ can correctly learn the knowledge between words and operators. In addition, we can see that CogSolver+ forms higher relation strength for reasonable knowledge than CogSolver (e.g., $wo_{i,c}$ of "total"-"+" is 0.644 and 0.313 for CogSolver+ and CogSolver, respectively), while a lower strength for invalid knowledge (e.g., $wo_{i,c}$ of "mins"-"×" is 0.025 and 0.115, respectively). This is attributed to our proposed knowledge recall mechanism, validating that it can effectively strengthens reasonable knowledge and weakens the unreasonable one.

E. Analysis on Knowledge Recall Mechanism

The key component of CogSolver+'s knowledge recall mechanism is our devised approximate estimation of $L(D_{re}, K', \hat{\theta}')$ in (21), which lays the foundation for consolidating knowledge given the replayed problems. It is motivated by our Theorem 1 proven in Section V-B, which asserts that the complexity required to accurately estimate $\hat{\theta}'$ is prohibitively high. In this section, we analyze how the approximation strikes a balance between estimation efficiency and effectiveness.

Specifically, we first discuss the impacts of the two most critical hyperparameters: r and γ in Algorithm 1. r represents the number of selected samples from D_{re} to unbiasedly estimate $\nabla_\theta L(D_{re}, K, \hat{\theta})^T \cdot H_{\hat{\theta}}^{-1}$, while γ controls the frequency of estimations to reduce the variance. We evaluate the performances of $r = \{1, 5, 10, 50, 100\}$ and $\gamma = \{1, 5, 10, 50, 100\}$ on Math23K in Fig. 10, using answer accuracy and the average running time of learning iterations (including a *Store-Apply-Update* process and the knowledge *Recall* mechanism) as the evaluation metrics (results on MAWPS can be found in Section 3 of Supplementary Material). To facilitate a more sufficient investigation, we consider both non-pre-train (i.e., CogSolver+) and pre-train scenarios (i.e., CogSolver+(B)). From Fig. 10(a), we can see that beyond $r \geq 5$, the answer accuracy exhibits marginal improvement with varying r, while the running time substantially increases by approximately $8\% \sim 50\%$. Comparatively, in Fig. 10(b), the accuracy reaches the peak at around $\gamma = 10$, with no significant difference in running time for $\gamma \in \{10, 50, 100\}$. Thus, under implementation setup $r = 5, \gamma = 10$, our knowledge recall mechanism strikes an optimal trade-off between effectiveness and efficiency, maintaining high accuracy in as less running time as possible. In addition, we observe that the running time of CogSolver+(B) is not much higher than that of CogSolver+, even though CogSolver+(B) comprises more parameters. It confirms that our approximation successfully reduces the time complexity with respect to the number of parameters, corroborating our proof in Section V-B.

To further emphasize the superiority of our approach, we design a variant named CogSolver+(O), which directly estimates $\hat{\theta}'$ through an additional Adam-based optimization process and then evaluates $L(D_{re}, K', \hat{\theta}')$ without any approximation. In Tables VI and VII, we compare its performances with CogSolver+, setting the number of optimization epoches (i.e., T in Section V-B) to $\{1, 10, 50, 100\}$, and conclude several key

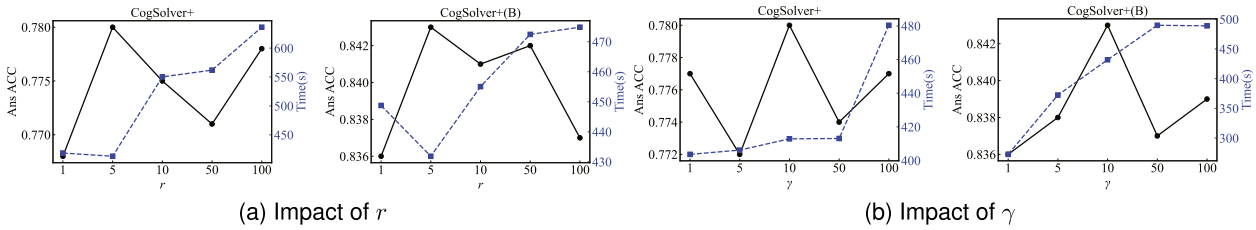


Fig. 10. The impacts of r and γ in Algorithm 1 on answer accuracy and average running time of learning iteration on Math23K.

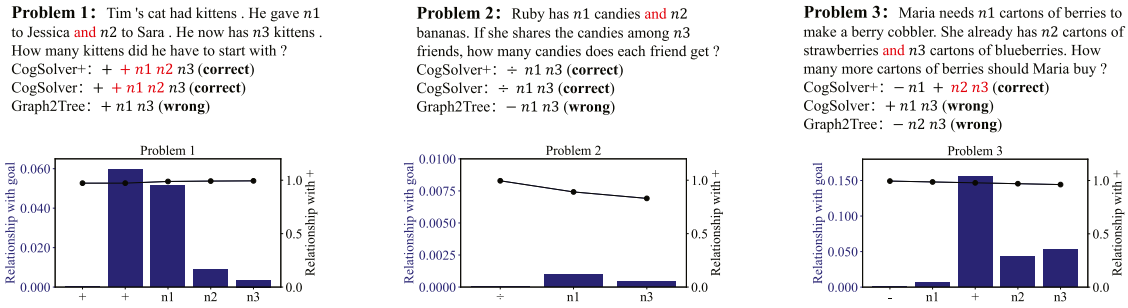


Fig. 11. Three cases of reasoning processes of CogSolver+. The x-axis presents the expression generation process. The black line and blue bar represent the relationship between "and" and "+", "and" and reasoning goal q_t at each reasoning step t, respectively.

TABLE VII
PERFORMANCE COMPARISON BETWEEN COGSOLVER+ AND COGSOLVER+(O) ON MAWPS

	CogSolver+	CogSolver+(O)			
		$T=1$	$T=10$	$T=50$	$T=100$
Ans ACC	0.838	0.829	0.833	0.832	0.840
Time(s)	79.87	152.31	468.25	1597.73	4047.59

observations. Firstly, the performance of CogSolver+(O) exhibits a positive correlation with the increase of T, indicating that an enhanced estimation of $\hat{\theta}'$ amplifies the effectiveness of knowledge recall mechanism. Secondly, CogSolver+(O) requires 2-10 times running time of CogSolver+ (even when $T=1$), but the performance difference is not substantial. This observation not only directly demonstrates the necessity of our proposed approximation method, but also strongly validates its efficacy in reducing the time complexity and accurately estimating the impact of updated knowledge. Furthermore, only when $T=100$ can CogSolver+(O) achieve a performance comparable to CogSolver+, which experimentally verifies the requirement of optimization time T proven in our Theorem 1.

F. Case Study

To illustrate how our CogSolver+ and CogSolver benefit from utilizing the learned knowledge to reason answers, we select three problems for case study in Fig. 11. We first report the descriptions of these problems and the predicted prefix expressions generated by our CogSolver+, CogSolver, and Graph2Tree. Notice that "and" appears in each problem, we then visualize the relationships between "and" and "+", "and" and goal at each reasoning step of CogSolver+.

From the black line, the relation strength between "and" and "+" is quite high ($wo_{i,c}^t \approx 1$) at the first reasoning step (i.e., $t=1$) for all problems. That is probably because "and" appears at a position that conveys the meaning of summation (sum of "kittens", "Ruby's", "berries"). Moreover, the strength remains high at $t=2,\ldots,5$ for problem 1 and 3 that need to do summation by "and" ($n1+n2, n2+n3$), while diminishing for problem 2 whose goal is irrelevant to summation. These results indicate that CogSolver+ can adaptively apply the knowledge ("and"-"+") based on contextual information, further affirming the importance and effectiveness of the knowledge-aware module.

The blue bar represents the relationship wg_i^t between "and" and the goal q_t, reflecting how focused the model is on "and" at each step t. For problem 1, it increases significantly when predicting "+" at $t=2$, which starts to complete $n1+n2$. A similar trend is observed at $t=3$ for problem 3, wherein the calculation of $n2+n3$ is undertaken. In contrast, the weight remains consistently low (<0.001) for problem 2, indicating that CogSolver+ considers "and" as irrelevant to the goal and disregard it at all steps.

Combining the above analysis, by paying more attention to "and" and relating it with "+", CogSolver+ correctly predicts "+" in problem 1 and 3. In contrast, Graph2Tree is unable to use such information and subsequently results in an error at step 2 and an incorrect answer. Meanwhile, CogSolver+ takes less into account "and" in problem 2, and therefore does not overestimate the probability of "+". In particular, we observe that CogSolver exhibits a similar performance to CogSolver+ in applying the knowledge of "and" and "+", but it still makes mistakes on problem 3. By scrutinizing its reasoning process, we find the reason is that CogSolver gradually forgets the knowledge between "berries" and "strawberries" during the learning process (the final $ww_{i,j}=0.621<\delta_1=0.7$), while CogSolver+ consolidates it by leveraging the knowledge recall mechanism ($ww_{i,j}=0.754$). As a result, CogSolver+ displays a superior understanding of the reasoning goal that computes "how many

berries there are" and correctly deduces "$+\ n2\ n3$" rather than "$+\ n1\ n3$". This exemplifies the crucial importance of our knowledge recall mechanism on the reasoning process. Based on these observations, we can conclude that our CogSolver+ has advantages in learning knowledge and applying it to different problems. Meanwhile, by integrating such knowledge within the goal-driven decoder, it accomplishes interpretable reasoning processes.

VII. Conclusion

In this article, we presented a focused study on enhancing machine intelligence in mathematical reasoning through autonomously learning knowledge. We first proposed a novel Cognitive Solver (CogSolver) by constructing two systems (*BRAIN-ARM*) and three steps (iterative *Store-Apply-Update*) to enumerate human-like cognitive behaviors inspired by *dual process theory* and *information processing theory*. Then, to tackle the problem of knowledge forgetting, we extended CogSolver to CogSolver+ by incorporating an essential knowledge *Recall* mechanism, drawing insight from another *complementary learning system theory*. From the experiments, we first demonstrated the effectiveness of CogSolver+ and CogSolver in answer reasoning for MWP. Then, we validated the rationality of both their learning results and processes, where we especially highlighted the superiority of CogSolver+. Furthermore, we provided an in-depth analysis of CogSolver+'s knowledge *Recall* mechanism, delineating how it struck a balance between estimation efficiency and efficacy. Finally, we showed how our models benefited from learned knowledge to achieve interpretable reasoning processes. In the future, there are several important research directions. First, our models are highly generalizable and extensible, and we will explore their potential in other domains. Second, to enhance the comprehensiveness, we can further incorporate knowledge from other fields and the implicit knowledge of large language models (LLMs). Third, there remain many other mechanisms of human cognition needed to be modeled in machines. For more discussions, please refer to Section 4 of Supplementary Material.

Acknowledgment

The authors are also very grateful to the anonymous reviewers for their constructive comments.

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] Z. Huang et al., "Neural mathematical solver with enhanced formula structure," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 1729–1732.

[3] A. Patel, S. Bhattamishra, and N. Goyal, "Are NLP models really able to solve simple math word problems," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol.*, 2021, pp. 2080–2094.

[4] Y. Wang, X. Liu, and S. Shi, "Deep neural solver for math word problems," in *Proc. Conf. Empirical Methods Natural Lang. Joint Conf. Natural Lang. Process.*, 2017, pp. 845–854.

[5] B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai, "ASTER: An attentional scene text recognizer with flexible rectification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, pp. 2035–2048, Sep. 2019.

[6] 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.

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

[8] Y. Cao, F. Hong, H. Li, and P. Luo, "A bottom-up dag structure extraction model for math word problems," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 39–46.

[9] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5149–5169, Sep. 2022.

[10] Z. Li et al., "Seeking patterns, not just memorizing procedures: Contrastive learning for solving math word problems," in *Proc. Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2022, pp. 2486–2496.

[11] X. Lin et al., "HMS: A hierarchical solver with dependency-enhanced understanding for math word problem," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 4232–4240.

[12] Z. Yang, J. Qin, J. Chen, L. Lin, and X. Liang, "Logicsolver: Towards interpretable math word problem solving with logical prompt-enhanced learning," in *Proc. Conf. Empirical Methods Natural Lang. Joint Conf. Natural Lang. Process.*, 2022, pp. 1–13.

[13] D. Guo, H. Wang, and M. Wang, "Context-aware graph inference with knowledge distillation for visual dialog," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6056–6073, Oct. 2022.

[14] Y. Zhang et al., "Skilful nowcasting of extreme precipitation with nowcastnet," *Nature*, vol. 619, no. 7970, pp. 526–532, 2023.

[15] G. Fei, S. Wang, and B. Liu, "Learning cumulatively to become more knowledgeable," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1565–1574.

[16] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 2, pp. 423–443, Feb. 2019.

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

[18] A. Lieto, D. P. Radicioni, and V. Rho, "Dual peccs: A cognitive system for conceptual representation and categorization," *J. Exp. Theor. Artif. Intell.*, vol. 29, no. 2, pp. 433–452, 2017.

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

[20] R. C. Atkinson and R. M. Shiffrin, "Human memory: A proposed system and its control processes," in *Psychology of Learning and Motivation*, vol. 2. New York, NY, USA: Elsevier, 1968, pp. 89–195.

[21] N. Çeliköz, Y. Erisen, and M. Sahin, "Cognitive learning theories with emphasis on latent learning, gestalt and information processing theories," *Online Submission*, vol. 9, no. 3, pp. 18–33, 2019.

[22] H. A. Simon, "Information-processing theory of human problem solving," in *Handbook of Learning and Cognitive Processes*, vol. 5. Oxfordshire, U.K.: Psychol. Press, 1978, pp. 271–295.

[23] 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.

[24] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, "Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory," *Psychol. Rev.*, vol. 102, no. 3, 1995, Art. no. 419.

[25] D. Kumaran, D. Hassabis, and J. L. McClelland, "What learning systems do intelligent agents need? Complementary learning systems theory updated," *Trends Cogn. Sci.*, vol. 20, no. 7, pp. 512–534, 2016.

[26] R. C. O'Reilly, R. Bhattacharyya, M. D. Howard, and N. Ketz, "Complementary learning systems," *Cogn. Sci.*, vol. 38, no. 6, pp. 1229–1248, 2014.

[27] F. Zhou and C. Cao, "Overcoming catastrophic forgetting in graph neural networks with experience replay," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 4714–4722.

[28] M. Irfan, Z. Jiangbin, M. Iqbal, Z. Masood, and M. H. Arif, "Knowledge extraction and retention based continual learning by using convolutional autoencoder-based learning classifier system," *Inf. Sci.*, vol. 591, pp. 287–305, 2022.

[29] Z. Liang et al., "Generalizing math word problem solvers via solution diversification," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 13183–13191.

[30] P. Lu, L. Qiu, W. Yu, S. Welleck, and K.-W. Chang, "A survey of deep learning for mathematical reasoning," in *Proc. Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2023, pp. 14605–14631.
[31] Y. Bakman, "Robust understanding of word problems with extraneous information," 2007, *arXiv:math/0701393*.
[32] S. Roy and D. Roth, "Solving general arithmetic word problems," in *Proc. Conf. Empir. Methods Natural Lang. Joint Conf. Natural Lang. Process.*, 2015, pp. 1743–1752.
[33] S. Shi, Y. Wang, C.-Y. Lin, X. Liu, and Y. Rui, "Automatically solving number word problems by semantic parsing and reasoning," in *Proc. Conf. Empirical Methods Natural Lang. Joint Conf. Natural Lang. Process.*, 2015, pp. 1132–1142.
[34] J. Qin, X. Liang, Y. Hong, J. Tang, and L. Lin, "Neural-symbolic solver for math word problems with auxiliary tasks," in *Proc. Conf. Empirical Methods Natural Lang. Joint Conf. Natural Lang. Process.*, 2021, pp. 5870–5881.
[35] Q. Wu, Q. Zhang, and Z. Wei, "An edge-enhanced hierarchical graph-to-tree network for math word problem solving," in *Proc. Conf. Empirical Methods Natural Lang. Joint Conf. Natural Lang. Process.*, 2021, pp. 1473–1482.
[36] 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, Hum. Lang. Technol.*, 2022, pp. 5944–5955.
[37] D. Huang et al., "Neural math word problem solver with reinforcement learning," in *Proc. 27th Int. Conf. Comput. Linguistics*, 2018, pp. 213–223.
[38] 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. Empirical Methods Natural Lang. Joint Conf. Natural Lang. Process.*, 2020, pp. 7137–7146.
[39] J. Liu et al., "Guiding mathematical reasoning via mastering commonsense formula knowledge," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2023, pp. 1477–1488.
[40] Z. Liang et al., "Mwp-bert: Numeracy-augmented pre-training for math word problem solvsing," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol.*, 2022, pp. 997–1009.
[41] W. Yu, Y. Wen, F. Zheng, and N. Xiao, "Improving math word problems with pre-trained knowledge and hierarchical reasoning," in *Proc. Conf. Empirical Methods Natural Lang. Joint Conf. Natural Lang. Process.*, 2021, pp. 3384–3394.
[42] J. Shen et al., "Generate & rank: A multi-task framework for math word problems," in *Proc. Conf. Empirical Methods Natural Lang. Joint Conf. Natural Lang. Process.*, 2021, pp. 2269–2279.
[43] 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, Hum. Lang. Technol.*, 2023, pp. 6858–6877.
[44] X. Zhu et al., "Solving math word problems via cooperative reasoning induced language models," in *Proc. Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2023, pp. 4471–4485.
[45] W. Chen, X. Ma, X. Wang, and W. W. Cohen, "Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks," *Trans. Mach. Learn. Res.*, 2023.
[46] P. Lu et al., "Inter-GPS: Interpretable geometry problem solving with formal language and symbolic reasoning," in *Proc. Conf. Empirical Methods Natural Lang. Joint Conf. Natural Lang. Process.*, 2021, pp. 6774–6786.
[47] A. Davies et al., "Advancing mathematics by guiding human intuition with AI," *Nature*, vol. 600, no. 7887, pp. 70–74, 2021.
[48] M. Bloch, "Language, anthropology and cognitive science," *Man*, vol. 26, no. 2, pp. 183–198, 1991.
[49] M. Ding, C. Zhou, Q. Chen, H. Yang, and J. Tang, "Cognitive graph for multi-hop reading comprehension at scale," in *Proc. Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2019, pp. 2694–2703.
[50] 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.
[51] K. A. Norman and R. C. O'Reilly, "Modeling hippocampal and neocortical contributions to recognition memory: A complementary-learning-systems approach," *Psychol. Rev.*, vol. 110, no. 4, 2003, Art. no. 611.
[52] D. Gao, R. Wang, S. Shan, and X. Chen, "CRIC: A VQA dataset for compositional reasoning on vision and commonsense," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5561–5578, May 2023.
[53] R. Navigli and M. Lapata, "An experimental study of graph connectivity for unsupervised word sense disambiguation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 678–692, Apr. 2010.
[54] H. Pei et al., "Multi-track message passing: Tackling oversmoothing and oversquashing in graph learning via preventing heterophily mixing," in *Proc. Int. Conf. Mach. Learn.*, 2024, pp. 40078–40091.
[55] J. D.M.-W. C. Kenton, and L. K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol.*, 2019, pp. 4171–4186.
[56] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proc. Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2014, pp. 55–60.
[57] W. Wang, T. Zhou, S. Qi, J. Shen, and S.-C. Zhu, "Hierarchical human semantic parsing with comprehensive part-relation modeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3508–3522, Jul. 2022.
[58] H. Ebbinghaus, "Memory: A contribution to experimental psychology," *Ann. Neurosciences*, vol. 20, no. 4, 2013, Art. no. 155.
[59] G. M. Van de Ven, H. T. Siegelmann, and A. S. Tolias, "Brain-inspired replay for continual learning with artificial neural networks," *Nature Commun.*, vol. 11, no. 1, 2020, Art. no. 4069.
[60] T. Korbak, H. Elsahar, G. Kruszewski, and M. Dymetman, "Controlling conditional language models without catastrophic forgetting," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 11499–11528.
[61] L. Averell and A. Heathcote, "The form of the forgetting curve and the fate of memories," *J. Math. Psychol.*, vol. 55, no. 1, pp. 25–35, 2011.
[62] H. Zhao et al., "An ensemble learning approach with gradient resampling for class-imbalance problems," *INFORMS J. Comput.*, vol. 35, no. 4, pp. 747–763, 2023.
[63] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[64] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1885–1894.
[65] R. D. Cook and S. Weisberg, *Residuals and Influence in Regression*. New York, NY, USA: Chapman Hall, 1982.
[66] R. Koncel-Kedziorski et al., "Mawps: A math word problem repository," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol.*, 2016, pp. 1152–1157.
[67] K. Cobbe et al., "Training verifiers to solve math word problems," 2021, *arXiv:2110.14168*.
[68] L. Wang et al., "Translating a math word problem to a expression tree," in *Proc. Conf. Empirical Methods Natural Lang. Joint Conf. Natural Lang. Process.*, 2018, pp. 1064–1069.
[69] L. Wang et al., "Template-based math word problem solvers with recursive neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 7144–7151.
[70] J. Li et al., "Modeling intra-relation in math word problems with different functional multi-head attentions," in *Proc. Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2019, pp. 6162–6167.
[71] B. Wang et al., "Structure-unified m-tree coding solver for math word problem," in *Proc. Conf. Empirical Methods Natural Lang. Joint Conf. Natural Lang. Process.*, 2022, pp. 8122–8132.
[72] 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.

Jiayu Liu received the BS degree in applied mathematics in 2020 from the University of Science and Technology of China (USTC), where he is currently working toward the PhD degree with the School of Artificial Intelligence and Data Science. He has authored or coauthored papers in refereed conference proceedings, such as ICML 2025, NeurIPS 2024, KDD 2023, and AAAI 2023. His research interests include data mining, knowledge representation and learning, and mathematical reasoning.

Zhenya Huang (Associate 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. He has authored or coauthored more than 50 papers in refereed journals and conference proceedings, including *IEEE Transactions on Knowledge and Data Engineering*, TOIS, *IEEE Transactions on Neural Networks and Learning Systems*, AAAI, KDD, SIGIR, and ICDM. His research interests include artificial intelligence, knowledge reasoning, and intelligent education. Dr. Huang has served regularly on the program committee of numerous conferences. He is a reviewer of leading academic journals.

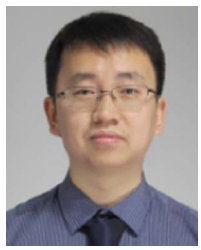
Enhong Chen (Fellow, IEEE) received the PhD degree from the University of Science and Technology of China (USTC), in 1996. He is currently a professor and vice director with the State Key Laboratory of Cognitive Intelligence. His research is supported by the National Science Foundation for Distinguished Young Scholars of China. He has authored or coauthored more than 200 papers in refereed conferences and journals, including *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Neural Networks and Learning Systems*, TOIS, ICML, NeurIPS, KDD, ICLR and AAAI. His research interests include data mining, machine learning, and artificial intelligence. He is an associate editor for *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Systems, Man, and Cybernetics*, ACM TIST, and WWWJ. He has served regularly on the organization and program committees of numerous conferences, including the program co-chair of ICKG-2020 and PAKDD-2022. Dr. Chen was the recipient of Best Application Paper Award on KDD 2008, Best Research Paper Award on ICDM 2011, Best Student Paper Award on KDD 2018 (Research), and Best Student Paper Award on KDD 2024 (Research).

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 research is supported by the National Science Fund for Excellent Young Scholars and Youth Innovation Promotion Association of Chinese Academy of Sciences. He has authored or coauthored more than 100 papers in refereed journals and conference proceedings, such as *IEEE Transactions on Knowledge and Data Engineering*, TOIS, *IEEE Transactions on Neural Networks and Learning Systems*, NeurIPS, ICML, ICLR, AAAI, and KDD. His research interests include data mining, knowledge discovery, and artificial intelligence. He has served regularly in the program committee of numerous conferences. He is a reviewer of leading academic journals. Dr. Liu was the recipient of KDD 2018 Best Student Paper Award (Research), ICDM 2011 Best Research Paper Award, and Alibaba DAMO Academy Young Fellow.

Hongke Zhao received the PhD degree from the University of Science and Technology of China, in 2018. He is currently an associate professor with Tianjin University. He has authored or coauthored more than 80 papers in refereed journals and conference proceedings, such as JOC, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Systems, Man, and Cybernetics*, *IEEE Transactions on Neural Networks and Learning Systems*, TEC, TOIS, KDD, AAAI, and IJCAI. His research interests include data mining, data-driven management, and algorithm management. He has served regularly in the program committee of numerous conferences. He is a reviewer of leading academic journals. Dr. Zhao was the recipient of the Distinguished Dissertation Award Nomination of Chinese Association for Artificial Intelligence (CAAI) in 2019.

Xin Lin received the BE degree in 2019 from the University of Science and Technology of China (USTC), where he is currently working toward the PhD degree with the School of Computer Science and Technology. He has authored or coauthored papers in refereed journals and conference proceedings, such as *IEEE Transactions on Neural Networks and Learning Systems*, AAAI 2021, and KDD 2021. His research interests include artificial intelligence and natural language processing.

Jing Sha is currently working toward the PhD degree in electronics and information with the University of Science and Technology of China (USTC). He is also with IFLYTEK Al Research. He has authored or coauthored papers in refereed conference proceedings, such as KDD and ACL. His research interests include natural language processing and intelligent education.



Shijin Wang received the PhD degree from the Institute of Automation, Chinese Academy of Science. He is currently the vice president with IFLYTEK Company, Ltd. and president with IFLYTEK AI Research (Central China). He has authored or coauthored more than 60 papers in refereed conferences such as ACL, KDD, and AAAI. His research interests include speech and natural language processing. He led the team that won more than ten championships in international technical evaluation, such as Blizzard Challenge and CHiME.