

Magic Sequence: LLM evaluation methodology of logical problem solving for sequence manipulation. Application to mathematical magic tricks.

Hugues ALI MEHENNI *Talan*
Paris, France
hugues.ali-mehenni@talan.com

Laurent CERVONI *Talan*
Paris, France
laurent.cervoni@talan.com

Abstract—This article explores the intersection between AI and magic through the analysis, resolution, and construction of mathematical magic tricks. The objective of this paper is to introduce a novel framework—comprising primarily previously unseen logical problems—for evaluating the reasoning capabilities of LLMs on sequence manipulation tasks. Through a review of existing literature and a series of experiments, we then discuss the ability of LLMs to explain and apply mathematical principles in magic, highlighting both the current potential and limitations of AI in this context. Our results reveal a partial knowledge and often misinterpretation of sequence manipulation operations by LLMs, highlighting the challenges associated with the abstraction and creativity required for magic

Index Terms—LLM evaluation, symbolic hybrid, logical problem solving, magical arts

I. INTRODUCTION

Since the introduction of the transformer architecture [1], which addressed the limitations of earlier seq2seq models, attention-based models have become the standard for most Natural Language Processing (NLP) tasks. By allowing increasingly large token windows to be kept in context, Large Language Models (LLMs) achieve impressive performance on long-sequence processing, continuously approaching human-level performance [2]. Nevertheless, these models still face a number of challenges when required to perform logical reasoning. First, their statistical nature does not guarantee the coherence of the different reasoning steps or the truthfulness of their inferences. Moreover, the autoregressive nature of LLMs tends to amplify errors as they predict subsequent tokens [3]. Finally, unlike some recent approaches [4], LLMs do not manipulate concepts but rather tokens, whose size is usually smaller than a word. Although they have been pretrained on vast amounts of text and code, one may question their analytical performance when dealing with less common sequence manipulations. It is from this perspective that this article focuses on mathematical magic tricks. While other domains of application are possible, this one has the advantage of being inherently little-known, incorporating artistic and psychological dimensions, and involving non-trivial sequence operations. Designed to stage surprising effects that challenge human common sense, understanding magic tricks thus

emerges as a comprehensive and complex evaluation task for an LLM.

While AI is now widely used in many arts (writing, image, audiovisual), magic remains perhaps one of the most “resistant” to AI due to its complexity. From its earliest historical mentions in the Westcar Papyrus (around 2000 BCE) to our digital age, magic has always sought to integrate—and even anticipate—the technological advances of its era. The use of AI and computational systems to assist in the design of new magic tricks has notably been explored in [5] and [6]. Focusing on tricks based on mathematical principles, the authors emphasize the necessity of maximizing and evaluating the magical impact of the trick [5]. Indeed, the effectiveness of a magic trick heavily relies on specific psychological constraints, such as perceptual errors and illusions [7], or misdirection and attention [8], whose roles in magic have been analyzed in [9]. In [10], authors also examine the specific impact of generative AI on the magical arts from the perspective of Baudrillard’s theory of hyperreality, highlighting how AI can both recognize and create magic, thereby posing new challenges for artists and audiences alike. With regard to visual perception, [11] compared the performance of a deep neural network to that of a human when presented with visual sleight-of-hand tricks, with the aim of identifying under what circumstances each would be fooled. While their work helps highlight some of our own cognitive biases exploited by magic tricks, it is primarily focused on the detection of manipulation-based tricks (sleight-of-hand).

The present study is more concerned with exploring the reasoning abilities of LLMs in the analysis, solution, and construction of magic tricks. In order to eliminate any bias stemming from visual perception, the magic tricks considered here are limited to mathematical tricks that can be fully characterized through a textual description. Particular attention has been given to the Gilbreath principle, as it involves notions of randomness and sequence permutations, and thus can represent a significant analytical challenge for an LLM.

Mathematical magic tricks thus represent a domain that is often multidisciplinary and inherently perplexing, incorporating psychological, cognitive, artistic, and increasingly digital elements. These questions more broadly reflect an approach

aimed at deepening our understanding of the potential interactions between AI and human creativity in a field that remains relatively untouched—one often characterized by a certain confidentiality of data among magicians, who are generally reluctant to reveal secrets to a lay audience (a practice known as “exposure”¹).

II. STATE OF THE ART

A literature review on logical problem solving with LLMs is presented below, which served to guide our approach. We then mention some existing benchmarks for these tasks (see Section II-B). Finally, although other mathematical techniques are explored in this study, one of the key concepts highlighted here is the Gilbreath principle, described in Section II-C.

A. Logical Reasoning in LLMs

Despite their growing success in various NLP tasks, LLMs remain less effective in logic and problem-solving tasks, mainly because such tasks do not tolerate any errors. Many recent studies have addressed this issue in the context of multi-step reasoning problems, often through ‘Chain of Thought’ mechanisms [12] [13]. For example, in [14], the model is encouraged to ‘take a step back’ through few-shot prompting which instructs the LLM to first ask a higher-level abstraction question, thereby anchoring its answer on more relevant facts. Conversely, in [15], the authors introduce a natural language plan search algorithm designed to promote greater diversity in the outputs generated by the LLM when producing solutions. Other works focus on guiding the sequence of steps in mathematical reasoning with tree search algorithms, such as in [16].

Since LLMs, by their nature, do not guarantee the truthfulness or consistency of reasoning, their use often relies on generating multiple candidate solutions, and may require an external deterministic tool to evaluate and verify both the sequence of reasoning and the correctness of the result. In this study, we focus on hybrid AI, and more specifically on neurosymbolic AI that combines a neural language model with a symbolic engine. This approach is motivated by various articles in the literature: for example, the idea of decoupling reasoning tasks from linguistic tasks [17], or dual-process theories of thinking [18] [19].

This method has yielded promising results with Alphageometry [20], which solves Olympiad-level geometry problems by using a symbolic engine to evaluate the state of a proof, and a large pre-trained language model to guide its exploration by suggesting new relevant geometric points and elements to add. A similar approach for quantitative mathematical reasoning was proposed in [21], where informal solutions are self-formalized into Isabelle code (a formal theorem-proving environment) to eliminate inconsistent solutions.

Even without the same guarantees as a theorem prover, another approach consists in formalizing some steps of mathematical problem solving using a programming language.

The logic programming language Prolog [22] offers several advantages in this regard. It is a high-level symbolic language designed for rule-based reasoning, operating according to predicate logic (specifically Horn clauses). It offers several types of solution search, including depth-first search, and allows for constraint programming. Thanks to its declarative programming paradigm (unlike imperative approaches such as Python), it emphasizes the relationship between declared elements, which more closely resembles natural language expression. One of the challenges of this approach, however, remains controlling the alignment between the formal language and the LLM’s natural language reasoning. In [3], authors thus propose incorporating natural language comments when generating Prolog code through fine-tuning.

B. Existing Benchmarks

In terms of datasets, numerous increasingly complex corpora have recently been developed to evaluate the mathematical reasoning abilities of LLMs in natural language:

- simple math problems described by situational text (“math-word” problems): GSM8k [23], SVAMP [24]
- purely mathematical problems: MATH [25], Frontier-Math [26], a challenging dataset that requires several hours of reflection for a researcher and on which LLMs achieve a success rate of only 2%.
- problems in the form of “Chain of Thought”, such as NuminaMath [27], one of the largest corpora collected so far (860,000 annotated examples).

More generally, benchmarks have also been proposed to evaluate the limitations of LLMs in solving logical problems [28].

C. Gilbreath Principle

The Gilbreath principle has been the subject of numerous publications since its introduction in [29] and is still widely used by magicians today. In the context of card magic, it involves using a pre-ordered deck of cards which, after dealing a portion of the deck onto the table and performing an American shuffle (or riffle shuffle) with the two resulting packets, retains certain block-ordering properties, even though the cards within a given block are no longer in their original order.

To understand the Gilbreath principle, it is first necessary to correctly understand what an American shuffle” is. This is defined as cutting a deck into two packets on the table and interleaving them” [30]. This type of shuffle has been extensively studied, for example, to determine the number of shuffles needed to achieve a random arrangement or to model it mathematically. In magic, the important aspect is that the spectator can see the cards being shuffled, and can even shuffle them themselves (which enhances the magical effect). From an algorithmic perspective, the shuffling of cards can be described by the Gilbert-Shannon-Reeds model [31].

Although several variants of the Gilbreath principle have been introduced over the years, the general mechanics can be summarized by a set of mathematical properties described by

¹the disclosure of secrets to a non-magician audience.

[32] and based on permutations. Intuitively, although several riffle shuffles can randomize the order of a deck, performing only a single shuffle preserves certain properties of the deck, because the two interleaved parts of the deck each retain their internal order, even though they are interleaved in the new deck. In [32], it is showed that a sequence with the periodic property of length p will satisfy the Gilbreath principle, meaning it will retain this property after any riffle shuffle, in every window of size p starting at a multiple of p , if this property is strongly p -compatible.

For the sake of simplicity and computational efficiency, we will limit ourselves to cases where this property describes individual characteristics of elements within each period (such as a partition of elements, e.g., n red cards per period). This case, in fact, corresponds to the vast majority of magic tricks based on the Gilbreath principle.

III. METHODOLOGY AND DATASET CONSTRUCTION

A. Formalism

In general, the explanation of a mathematical magic trick could be modeled as the search for a formal proof that the trick works, similar to a theorem prover. However, these present a number of additional challenges:

- **Complex random manipulations:** Many magic tricks involve random shuffling of cards, especially those using the Gilbreath principle. For a riffle shuffle, the number of possible shuffles is exponential in the sequence length n , resulting in an exponential number of solutions (2^{n-1} permutations). Unlike typical annotated mathematics problems (see benchmarks section II-B), it is not possible here to evaluate the LLM's output based on a fixed answer. A symbolic tool (here, a Prolog rule system) is therefore necessary to verify the correctness of the found solution. To reduce computational cost, we restricted ourselves to cases where Gilbreath properties correspond to individual elements (partition of the initial set).
- **An underexplored domain:** Unlike other mathematical datasets, the domain of magic tricks is relatively uncontaminated. While classic tricks may be known to LLMs, they are unlikely to have been exposed to their variants, and some of the tricks presented here still benefit from a degree of professional secrecy.
- **An artistic and psychological component:** As indicated in section ??, these components would be difficult to apprehend using only formal rules. For this reason, we chose to keep a natural language statement for all these tricks. However, this raises the question of alignment between natural language and formal language (Prolog), particularly for the automatic generation of tricks. Training a language model to generate informal solutions requires large amounts of data, which are difficult to obtain in the field of magical arts. Dataset construction therefore relied more heavily on the use of formal rules (see next section).

The evaluation task proposed in this study is thus to explain a magic trick based on its description and according to different levels of contextual information. Evaluation on the proposed dataset is then performed on two elements:

- 1) the identification of the concept associated with the trick;
- 2) the verification of the solution proposed by the LLM dynamically using a symbolic tool. This solution must satisfy the constraints of the Prolog rules associated with the trick. Concretely, this amounts to determining an initial sequence that works for this trick.

B. Construction

A first dataset of mathematical magic tricks was initially collected to serve as a reference for generating other tricks. This initial dataset includes about half a dozen magic tricks that feature two key concepts explaining their operation:

- **the Gilbreath principle** (see section II-C)
- **the 14/15 forcing:** tricks where the sum of the values of two adjacent cards takes only a limited number of values for an initially ordered deck—a mechanism first described by Medrington in 1917.²

These reference tricks feature a natural language statement, which is then aligned with a sequence of Prolog operations (Prolog code, see Appendix). To these basic rules are added Prolog rules specific to the type of sequence used for the trick (for example, a 52-card deck). Each “deck” is then represented as a finite domain of integer elements so as to be able to apply constraint logic programming principles (clpfd library in Prolog) and thus optimize solution search. A 52-card deck is thus interpreted as a list of distinct integers from 1 to 52.

A (not necessarily unique) solution for each trick can thus be obtained using Prolog software in finite time, thanks to the use of simplification rules, especially designed to reduce the computational cost of operations such as random shuffles.

To artificially augment this database, some variations in the Prolog rules were introduced. Although generation via LLM was considered initially, the absence of a fine-tuned model led to inconsistent and poorly aligned tricks. These variations were therefore generated from deterministic formal rules ensuring the accuracy of the tricks. We can then distinguish three types of variations introduced to the existing tricks (see Figure I):

- Variation in the type of sequence: 5 different decks were used (52-card deck, 32-card deck, Cluedo cards, uppercase and lowercase letters, dominoes)
- Variations in sequence length and in the properties of their elements (predictions on red cards, face cards, sum value, etc.)
- Addition of noise via operations/concepts that do not alter the functioning of the trick (cuts, etc.)

The resulting list of operations is then verified using Prolog software. If the trick does converge to a solution, it is added to the database. The result is a dataset in the form of a list of magic trick operations. An alignment between Prolog operations and a natural language description is then performed

²https://www.conjuringcredits.com/doku.php?id=cards:14_15_stack

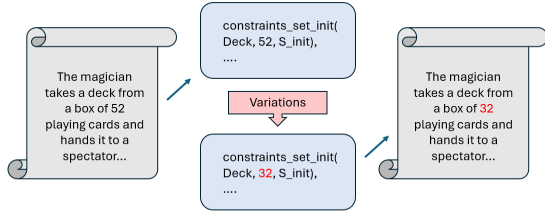


Fig. 1. Process of variation and verification of generated tricks: Tricks are converted into Prolog operations, then deterministic variations are introduced into the code, which is executed to check for convergence. Finally, natural language alignment is performed via rules.

TABLE I
KNOWLEDGE AND IDENTIFICATION (ID.) RATE OF THE TWO USED CONCEPTS (FORCING AND GILBREATH) FOR DIFFERENT LLMs, PASS@1 WITH 50 VARIATIONS (SAMPLES) PER MAGIC TRICK

LLM	GPT-4O	GPT-4.1	LeChat	Claude.ai
Gilbreath knowledge	✓	✓	✓	✓
Forcing knowledge	✓	✓	×	✓
Gilbreath Id. rate (classic)	0.27	1.0	1.0	1.0
Gilbreath Id. rate (all)	0.09	1.0	-	-

using deterministic rules in order to normalize the descriptions and ensure the accuracy of the generated tricks.

IV. RESULTS

A. Knowledge of the Principles Used

The use of these two concepts allows for a comparison between distinct behaviors of LLMs: while the Gilbreath principle is generally recognized by LLMs, the 14/15 forcing remains largely unfamiliar to them a priori (see Table I).

The Gilbreath principle is particularly well known in the following setup: a 52-card deck with alternating red and black cards, which retains pairs of different colors after a riffle shuffle, here referred to as the “classic trick.” The introduction of variations allows for a more precise assessment of the understanding of this principle. The objective is therefore to determine whether an LLM is capable of recognizing its use and grasping its application.

Table I shows that the Gilbreath principle is now consistently recognized by most state-of-the-art LLMs (GPT-4.1, Claude, etc.), even when it appears in new tricks (artificially created) and even for “packs” with unusual sequences, such as dominos or letters. This demonstrates their ability to recognize the principle outside the usual context and domain in which they have likely encountered it during training. The second principle, which has no open publications, is less recognized and is often confused with other known principles, in particular Gilbreath and Miraskill. Nevertheless, most models recognize the notion of forcing, though often only partially.

B. Understanding Sequence Manipulations

However, current models still encounter difficulties in correctly manipulating sequences to explain different magic tricks, even though they recognize the underlying principle.

This represents a significant area for improvement for these LLMs, not only in the context of mathematical magic tricks but also in any field involving mixing and multiple operations on sequences.

In particular, according to the results in Figure 2, GPT-4O tends to fail to find an initial sequence enabling the trick when:

- The trick involves an unusual packet (such as a sequence of dominos), even if the sequence length is short.
- The trick includes strong constraints on the choice of cards and/or a random selection of a number of cards by the spectator (complex Gilbreath trick and Forcing trick).

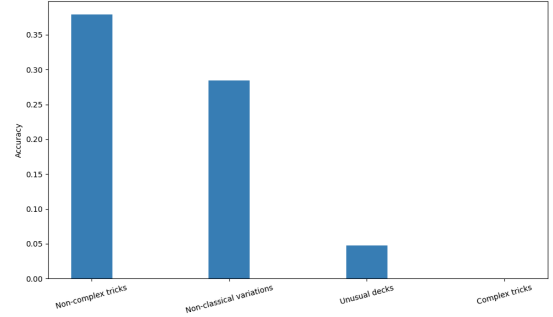


Fig. 2. Average success of GPT-4O (pass@1) on different variations of 4 Gilbreath tricks (1 classic, 2 non-classic and one complex) with 50 variations for each

Although GPT-4.1 achieves better performance on all non-complex Gilbreath tricks (81% accuracy), it also completely fails to explain complex Gilbreath tricks or forcing tricks that involve a random choice of the number of cards by the spectator and incorporate stronger constraints on the initial sequence. This result persists even when employing few-shot and chain-of-thought prompting. It corroborates recent findings in the literature on reasoning models, indicating that both LLMs and LRMs seem to experience a complete breakdown when confronted with tasks of high complexity [33].

V. DISCUSSION

This paper proposes a framework for constructing and evaluating novel magic tricks to analyze the capabilities of LLMs in sequence manipulations, with minimal data contamination. While the use of previously unseen magic tricks introduces an element of surprise in the evaluation of LLMs, an approach that could potentially be extended to other magical principles, a central strength of this methodology lies in its capacity to systematically generate controlled variations. As knowledge of common magic tricks becomes increasingly accessible, the integration of more intricate and less familiar variants may prove necessary, particularly given that complex tricks continue to challenge even state-of-the-art LLMs.

This initial investigation reveals notable limitations in the models’ ability to generalize abstract constructs such as mixing procedures and random cutting operations. While the direct application of Gilbreath or Forcing principles may be inherently constrained, the broader objective of enhancing LLMs’ reasoning and sequence manipulation capabilities holds promise for

improving their performance across a range of sophisticated tasks, including scientific research, data interpretation, and beyond.

One aspect that could be further explored is the generation of mathematical magic tricks by LLMs. In the context of quantifiable evaluation, we chose to strongly guide the construction of the tricks to avoid potential misalignment.

Another perspective would be to focus on the multimodal and particularly visual aspects of magic tricks, whether or not they involve manipulation. The explanation of the trick should then take into account the uncertainty in the operations and manipulations performed by the magician or the spectator.

APPENDIX

List of Prolog operations describing a mathematical card trick:

- `cutting(S1, S2)`: Performs a complete cut (cut and reattach) at a random position on the sequence `S`, resulting in the sequence `S2`. This corresponds to a random cut of a deck.
- `top_N(N, Sequence, Top, End)`: Retrieves the sequence `Top`, which corresponds to the first `N` elements of the sequence `Sequence`. The remainder of `Sequence` then becomes the sequence `End`. This operation, for example, corresponds to taking the top of a deck.
- `deal_N_for_M(N, M, Sequence, Hands, Rest)`: Deals the elements of the sequence `Sequence` one by one in order to `M` people, so that each person receives `N` elements. This produces a list `Hands` of `M` sequences of size `N`. The remainder of `Sequence` becomes the sequence `Rest`. For example, this corresponds to dealing `M` hands of `N` cards.
- `deal_all_for_M(M, Sequence, Hands)`: Deals the entire sequence `Sequence` one by one in order to `M` people as evenly as possible. This yields `Hands`, a list of sequences.
- `set_collect_tricks(Hands, Tricks)`: Gathers all elements one by one from each sequence in the list `Hands` into the sequence `Tricks`. This is the inverse operation of dealing.
- `set_collect_hands(Hands, S)`: Concatenates all sequences from `Hands` to form a sequence `S`. This operation, for example, corresponds to gathering all hands together to form a deck.
- `set_collect_ordered_hands(Hands, Index_Order, S)`: Concatenates all sequences from `Hands` into a sequence `S` following the order given by `Index_Order`.
- `get_hands_from_seq(N, S, Hands)`: Extracts `Hands` (a list of sequences of `N` elements) from the elements taken in order from a sequence `S`. This is the inverse operation of `set_collect_hands(Hands, S)`.
- `select_hand(C, Hands, N)`: Selects the hand number `N` in `Hands` that contains the element `C`.
- `riffle_shuffle(N, S1, S2, S)`: Performs a riffle shuffle (random interleaving) between the sequences `S1` and `S2`, producing the sequence `S` of length `N`.
- `partition_N(N, Period, Deck, Conditions, S)`: Finds a partition of `N` elements from a set `Deck` that results in a periodic sequence `S` satisfying the list of conditions `Conditions`, in the same order for each period of size `Period`. Each condition is a list `[Num, Cla]`, corresponding to the exact number `Num` of elements of class `Cla` present in each period.
- `constraints_set_init(Deck, N, S_init)`: Defines the constraints on the initial sequence `S_init` of size `N`, which must correspond to a partition of the set `Deck`.
- `constraints_apparent_set_init(Deck, N, S_init, Apparent)`: Defines the constraints on the initial sequence `S_init` of size close to the apparent size `Apparent`, corresponding to a partition of a set `Deck`.
- `constraints_specific_init(Deck, Properties, N, S_init)`: Defines the constraints on the initial sequence `S_init` of size `N`, composed only of elements from `Deck` satisfying at least one of the properties in `Properties`.
- `choose_N(N, Low, High)`: An integer is chosen by the spectator between the values `Low` and `High`, inclusive.

REFERENCES

- [1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.
- [2] C. R. Jones and B. K. Bergen, "Does gpt-4 pass the turing test?," 2024.
- [3] X. Tan, Y. Deng, X. Qiu, W. Xu, C. Qu, W. Chu, Y. Xu, and Y. Qi, "Thought-like-pro: Enhancing reasoning of large language models through self-driven prolog-based chain-of-thought," 2024.
- [4] L. Barrault, P.-A. Duquenne, M. Elbayad, A. Kozhevnikov, B. Alastruey, P. Andrews, M. Coria, G. Couairon, M. R. Costa-jussa, D. Dale, H. Elsahar, K. Heffernan, J. ao Maria Janeiro, T. Tran, C. Ropers, E. Sánchez, R. S. Roman, A. Mourachko, S. Saleem, and H. Schwenk, "Large concept models: Language modeling in a sentence representation space," 2024.
- [5] H. Williams and P. W. McOwan, "Magic in the machine: a computational magician's assistant," *Front. Psychol.*, 2014.
- [6] H. Williams and P. W. McOwan, "The magic words: Using computers to uncover mental associations for use in magic trick design," *PONE*, 2017.
- [7] J. Robinson, *The Psychology of Visual Illusion*. Mineola, NY: Dover Publications., 1998.
- [8] R. Rensink, J. O'Regan, and J. Clark, "To see or not see: the need for attention to perceive changes in scenes," *Psychological Science*, 8(5), 368- 373, 1997.
- [9] G. Kuhn, B. Tatler, J. M. Findlay, and G. G.Cole, "Misdirection in magic: implications for the relationship between eye gaze and attention," *Vis. Cogn.* 16, 391-405., 2008.
- [10] A. Allaire and L. Cervoni, "Magical simulacra and simulating magic: A baudrillardian perspective on the recognizing and creating magic using generative artificial intelligence," *The Polish Journal of Aesthetics*, 2023.
- [11] R. Zaghi-Lara, M. Gea, J. Camí, L. Martínez, and A. Gomez-Marín, "Playing magic tricks to deep neural networks untangles human deception," 2019.
- [12] A. Creswell, M. Shanahan, and I. Higgins, "Selection-inference: Exploiting large language models for interpretable logical reasoning," in *The Eleventh International Conference on Learning Representations*, 2023.

- [13] A. Saparov and H. He, “Language models are greedy reasoners: A systematic formal analysis of chain-of-thought,” in *The Eleventh International Conference on Learning Representations*, 2023.
- [14] H. S. Zheng, S. Mishra, X. Chen, H.-T. Cheng, E. H. Chi, Q. V. Le, and D. Zhou, “Take a step back: Evoking reasoning via abstraction in large language models,” *Google Deepmind*, 2024.
- [15] E. Wang, F. Cassano, C. Wu, Y. Bai, W. Song, V. Nath, Z. Han, S. Hendryx, S. Yue, and H. Zhang, “Planning in natural language improves llm search for code generation,” 2024.
- [16] A. Wang, L. Song, Y. Tian, B. Peng, D. Yu, H. Mi, J. Su, and D. Yu, “Litesearch: Efficacious tree search for llm,” 2024.
- [17] K. Mahowald, A. A. Ivanova, I. A. Blank, N. Kanwisher, J. B. Tenenbaum, and E. Fedorenko, “Dissociating language and thought in large language models: a cognitive perspective,” 2023.
- [18] G. Gronchi and A. Perini, “Dual-process theories of thought as potential architectures for developing neuro-symbolic ai models,” *Front. Cognit.*, 2024.
- [19] D. Kahneman, *Thinking, Fast and Slow*. 2012.
- [20] T. Trinh, Y. Wu, and Q. L. et al., “Solving olympiad geometry without human demonstrations,” *Nature* 625, 476–482, 2024.
- [21] P. Zhou, C. Staats, W. Li, C. Szegedy, K. Q. Weinberger, and Y. Wu, “Don’t trust: Verify – grounding llm quantitative reasoning with auto-formalization,” *ICLR*, 2024.
- [22] K. A. Bowen, “Prolog,” in *Proceedings of the 1979 Annual Conference, ACM ’79*, (New York, NY, USA), p. 14–23, Association for Computing Machinery, 1979.
- [23] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, “Training verifiers to solve math word problems,” 2021.
- [24] A. Patel, S. Bhattamishra, and N. Goyal, “Are NLP models really able to solve simple math word problems?,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Online), pp. 2080–2094, Association for Computational Linguistics, June 2021.
- [25] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, “Measuring mathematical problem solving with the math dataset,” 2021.
- [26] E. Glazer, E. Erdil, T. Besiroglu, D. Chicharro, E. Chen, A. Gunning, C. F. Olsson, J.-S. Denain, A. Ho, E. de Oliveira Santos, O. Järvinen, M. Barnett, R. Sandler, M. Vrzala, J. Sevilla, Q. Ren, E. Pratt, L. Levine, G. Barkley, N. Stewart, B. Grechuk, T. Grechuk, S. V. Enugandla, and M. Wildon, “Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai,” 2024.
- [27] J. Li, E. Beeching, L. Tunstall, B. Lipkin, R. Soletskyi, S. C. Huang, K. Rasul, L. Yu, A. Jiang, Z. Shen, Z. Qin, B. Dong, L. Zhou, Y. Fleureau, G. Lample, and S. Polu, “Numinamath.” [<https://github.com/project-numina/aimo-progress-prize>](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.
- [28] P. Giadikiaroglou, M. Lymperaoui, G. Filandrianos, and G. Stamou, “Puzzle solving using reasoning of large language models: A survey,” 2024.
- [29] N. Gilbreath, “Magnetic colors,” *The Linking Ring*, 38(5), p60., 1958.
- [30] P. Diaconis and R. Graham, “Magical mathematics: the mathematical ideas that animate great magic tricks,” *Princeton University Press.*, 2011.
- [31] D. Bayer and P. Diaconis, “Trailing the dovetail shuffle to its lair,” *Dovetail Shuffle to Its Lair: The Annals of Applied Probability*, 2, 294–313., 1992.
- [32] M. Cayrol, “Le principe de gilbreath, généralisation, variantes et magie des cartes,” *Quadrature*, 2015.
- [33] P. Shojaei, I. Mirzadeh, K. Alizadeh, M. Horton, S. Bengio, and M. Farajtabar, “The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity,” 2025.