Chain-of-Thoughts Prompting with Language Models for Accurate Math Problem-Solving

Sze Ching Evelyn Fung
Diocesan Girls' School
Hong Kong, China
e@evelynfung.com

Man Fai Wong
City University of Hong Kong
Hong Kong, China
mfwong29-c@my.cityu.edu.hk

Chee Wei Tan
Nanyang Technological University
Singapore
cheewei.tan@ntu.edu.sg

2023 IEEE MIT Undergraduate Research Technology Conference (URTC) | 979-8-3503-0965-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/URTC60662.2023.10534945

***Abstract*—Large Language Models (LLMs) have gained usage across various domains, especially in education. However, the current state-of-the-art LLMs fail in numerical calculations due to their reliance on the pre-trained dataset that does not focus on mathematical oversight. Prompting is crucial to guide LLMs to yield desired outputs for mathematical problems. This paper explores a new Chain-of-Thoughts (CoT) prompting framework, leveraging Python-based tools like LLM Math, LLM symbolic math, and SerpAPI. We also evaluate the existing works with the CoT prompting framework for math problem-solving. Students can utilize this framework to obtain more precise solutions and comprehensive explanations for their queries.**

***Index Terms*—Large Language Models, Prompt Engineering, LangChain, Mathematical reasoning, Conversational AI**

I. Introduction

Large Language Models (LLMs) is one of the most prominent technologies used nowadays. These advanced artificial intelligence (AI) models are designed to process and generate human-like text [1]. They undergo extensive training on massive datasets, incorporating vast collections of text from diverse sources. With their exceptional performance on multiple Natural Language Processing (NLP) datasets, LLMs demonstrate their prowess across a wide range of tasks, including translation [2], content generation [3], and conversational AI [4]. LLMs like GPT-3.5 and GPT-4, equipped with billions of parameters, excel in delivering highly accurate and context-aware responses. Among them, GPT-4 stands out as a new generation LLM [5], representing a significant advancement in the field with higher levels of general intelligence compared to its predecessors. However, it appears that LLMs do not perform satisfactorily when it comes to solving mathematical problems [6]. These models have shown poor performance in arithmetic reasoning tasks, struggling to generate accurate solutions for mathematical problems [7]. Unlike natural language understanding, mathematical problems usually have a definite and singular correct solution, making the task more challenging for LLMs.

Currently, researchers have been exploring different methods to improve the accuracy of solving mathematical problems. Some have employed few-shot prompting [8], while others have utilized the chain of thoughts technique in GPT-3.5 to enhance results [9]. Another intriguing approach involves GPT-3.5 generating Python code for solving math problems [10]. These diverse approaches hold promise in advancing the capabilities of LLMs in mathematical problem-solving. Existing tools for solving math problems often come with a monthly subscription, but they have limitations. They may struggle with mathematical word questions or riddles due to the lack of LLMs-based approaches. Additionally, these tools have fixed equation parameters tied to the initial question, limiting their ability to provide specific examples as requested by teachers or students [11]. This lack of flexibility makes them less user-friendly and interactive. In contrast, the integration of frameworks like Langchain has received less attention.

In this study, we propose a Chain-of-Thoughts (CoT) prompting framework using Langchain alongside GPT-3.5 to address math problems. The framework enables two-sided conversations between users and the LLM to enhance the accuracy and efficiency of mathematical problem-solving while mitigating issues like AI hallucination. Overall, the contributions can be summarized as follows:

- We propose a novel CoT prompting framework that leverages LangChain and LLMs to effectively utilize the Chain of Thoughts for solving mathematical problems.
- We present a comprehensive evaluation and comparison of the performance between our proposed framework and existing tools on the mathematical reasoning problems.
- Source codes and dataset are available on GitHub (https://github.com/EvelynFung/Langchain-Math-Model).

II. Related Works

A. Prompt Engineering

Prompting proves invaluable in leveraging LLMs in a natural and intuitive manner. However, model interpret prompts differently from humans, necessitating prompt engineering [12]. Effective prompt engineering entails providing the model with appropriate instructions, context, user input, and output indicators [13]. Communication between users and GPT-3.5, along with other LLMs, primarily occurs through prompts, which direct the model to produce responses aligned with the user's objective [14]. Understanding the intricacies of prompt engineering is crucial for developing meaningful interactions with these LLMs as the quality of prompts directly impacts the generated responses' quality.

Prompting techniques include zero-shot, few-shot, role and CoT prompting. Zero-shot prompting refers to the capability of a language model to generate responses or outputs for tasks or prompts it has never encountered during training [15]. Once the model comprehends the context of the task, it can produce appropriate responses accordingly. This approach is particularly effective for straightforward tasks like arithmetic calculations. Hence, in this study, zero-shot prompting will be utilized to leverage the language model's adaptability and proficiency in solving mathematical problems.

Few-shot prompting involves providing a limited number of task-specific training examples or prompt-response pairs to a LLM [16]. With this scarce training data, the model adapts its knowledge and enhances its performance on the particular task. This approach proves effective for tackling more complex tasks that cannot be straightforwardly addressed by a single prompt. In contrast to zero-shot prompting, which relies solely on understanding the task context, few-shot prompting allows the model to benefit from a small amount of targeted training, making it a valuable technique for handling intricate tasks.

The problem-solving skills of LLMs have been significantly enhanced by Chain of Thoughts (CoT) prompting, which can be either zero-shot or few-shot learning [17]. When humans calculate a math problem, they decompose the intermediate steps to arrive at the final solution. Similarly, LLMs can exhibit the ability to generate a coherent sequence of logical steps leading to a solution for a given problem [9], demonstrating a cohesive chain of reasoning.

B. Langchain Framework

LangChain [18] is a powerful framework designed for building applications with LLMs, effectively chaining different components together. The framework comprises three main components: LLM, Tools, and Agents, which work in tandem to control the interaction as shown in Figure 1. Although LLMs have their limitations in providing detailed responses for questions that require expertise, they excel in offering general responses within specific domains. To tackle this, LangChain introduces an innovative approach to handling prompts. For instance, in a GPT-3.5 model, an AgentExecutor chain can be established using the Python package LangChain. This chain empowers the framework to execute various tasks, such as running web searches, evaluating the results, and generating appropriate responses based on the acquired information. The process commences with the preprocessing of the corpus, dividing it into chunks or summaries, and converting them into a vector space. Subsequently, when a question is posed, LangChain agents search for similar chunks to identify the suitable actions and tools required for an accurate response. The final output is then conveyed back to the user through the LLM, ensuring a streamlined and efficient prompt answering process. With LangChain's capabilities, applications can leverage the strengths of LLMs while addressing their limitations, leading to more effective and dynamic interactions with users.

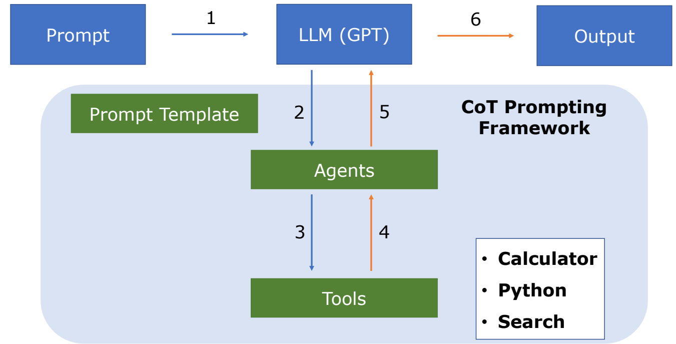

Fig. 1: Workflow for handling the prompts and outputs.

III. CoT Prompting Framework

A. Implementation

In this study, Langchain serves as the foundation for creating a framework to solve mathematical problems. The framework incorporates two powerful techniques: zero-shot learning and CoT. These techniques enhance the framework's ability to handle math problems effectively. To achieve this, various tools are integrated into the framework. For conducting web searches, Google Search is employed using the SerpAPI tool. Additionally, the framework leverages the LLM symbolic tool for symbolic math calculations, LLM Math for solving math problems, and Python for graphing curves.

Our framework is developed on top of GPT-3.5, exhibiting a distinct learning behavior compared to few-shot examples during runtime. Instead, it relies on effectively locating the task within its existing space of learned tasks [19]. This study aims to build on this success by focusing on further enhancing accuracy through the implementation of the zero-shot prompting method. By leveraging GPT-3.5's ability to understand and respond to prompts without explicit training, the study aims to explore how to fine-tune the framework's performance in mathematical problem-solving. The goal is to improve accuracy and expand the framework's capabilities in handling a broader range of math problems effectively.

B. Tools Integration with CoT Prompting Framework

Our framework utilizes SerpAPI, a real-time Application Programming Interface (API) that provides access to Google search results and other search engines based on user queries. By integrating SerpAPI with Langchain, our model gains the ability to browse the web and acquire relevant data dynamically. Integrating SerpAPI with Langchain enables the embedded LLMs to expand its knowledge beyond its initial training data. This dynamic combination empowers the framework to perform comprehensive web searches, allowing it to gather up-to-date information and insights on a wide range of topics. As a result, our LLMs can access real-time data and stay current with the ever-evolving information available across the web. This integration enhances the framework's versatility and effectiveness in generating accurate and contextually relevant responses to user queries.

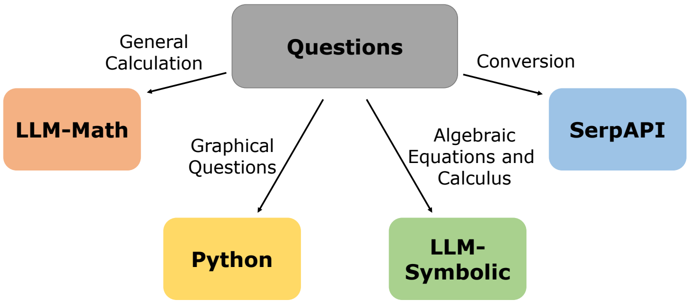

Fig. 2: Tools integrated on CoT prompting framework for each type of questions.

Langchain offers two math chains, LLM Math and LLM Symbolic Math, which excel in resolving complex word math problems and solving algebraic equations. Similar to humans, LLMs are proficient in writing programs but may face challenges in executing them, especially when dealing with difficult mathematical problems. To address this, a powerful solution involves combining an LLM with an appropriate program runtime. This process entails transforming unstructured language into a program representation using the LLM and then executing the resulting code with a more specialized and straightforward tool, such as a calculator or specific software. For example, an LLM can effortlessly translate complex mathematical issues into code [20], while a specialized mathematical computation tool, like calculators, can efficiently conduct the computation itself. This combination leverages the strengths of both the LLM and the runtime tool, enabling more effective and accurate mathematical problem-solving.

Given the 9 types of mathematical question categories, the agents within Langchain play a crucial role in determining which tools to utilize for each specific question type. Through rigorous testing and evaluation of the code, Figure 2 illustrates the decision-making process of Langchain in selecting the appropriate tool for handling each question type. This dynamic and intelligent approach enables Langchain to efficiently and accurately address a diverse array of mathematical queries, optimizing the use of tools based on the nature of the question at hand. The interplay between the agents and the tool selection process ensures that Langchain is well-equipped to handle various mathematical challenges, providing effective and contextually relevant solutions for each question type.

C. CoT Prompting Framework with Math Skills

In our framework, the GPT-3.5 served as the LLM, with a crucial temperature setting of 0. The choice of temperature significantly impacts the model's output, with high values resulting in more creative and unexpected responses, and low values leading to more conservative and predictable results [21]. By using a temperature of 0 for mathematical calculations, the model gains confidence in its predictions, providing more precise answers to the queries. To gain insights into the framework's thought process, the verbose option was set to true. This enables us to observe the CoT followed by the framework, offering valuable information for further analysis and improvement. Additionally, each tool in the framework is described in a manner that facilitates the agent's selection of the appropriate tool based on its description. This clarity ensures efficient utilization of tools for specific tasks, ultimately enhancing the framework's overall performance.

The `ConversationalReact` agent serves as the primary tool for handling mathematical problems, excluding graphical questions, within our framework. A React framework is utilized to provide both reasoning and action capabilities. Our framework employs suitable tools based on those thoughts. A key aspect of the framework is the significance of chat history in the conversation. As users interact with the system, they may ask additional questions related to the original math problem or seek further clarifications. The framework takes this conversational context into account to provide more coherent and accurate responses. To facilitate effective conversation and memory retention, it also employs various types of conversational memory, and in this specific framework, the `ConversationBufferMemory` is used to store and retrieve conversation details. By utilizing this memory type, the framework can store the entire conversation's context, allowing the model to maintain a comprehensive understanding of the ongoing discussion. This becomes particularly valuable when users wish to ask follow-up questions, as the LLM can still provide relevant responses with a continuous awareness of the conversation's context.

Our framework also uses a Python agent executor for graph-related questions, bridging the gap between the LLM and Python. This addresses the limitations of other components, which cannot effectively plot graphs or solve graphical questions. The Python agent executor enables the framework to execute Python code, create graphs, perform complex calculations, and derive insights from plotted graphs. When a graph-related question is received, the framework forwards it to the Python agent executor, which executes the Python code and returns the results to the LLM. After executing the code, it can be seen that the framework will solve the math questions using the Agent Executor chain. The framework's reasoning process is displayed transparently step by step. It utilizes the Agent Executor chain to navigate through a series of specialized tools, leading to the final solution.

IV. Evaluation

A. Dataset and Settings

In this study, DeepMind Mathematics Dataset [22] is used to evaluate the performance of our proposed framework. The dataset consists of math questions with various levels. Selected questions cover a diverse array of mathematics topics, thoughtfully integrated into a coherent learning curriculum. Topics encompass algebra, arithmetic, calculus, comparison, measurement, numbers, polynomials, and probability. From this dataset, 10 challenging questions were chosen for each topic, resulting in 80 questions. Additionally, 10 graphical questions were extracted from the International A-Levels. Together, this compilation offers a comprehensive and varied

Question	GPT-3.5 Response (Zero-shot, Chain of Thoughts)	CoT Prompting Framework's Response
Solve 1058*a - 271 - 2541 = 14*a - 362*a for a. (Algebra)	Combine like terms on the left side: 1058a - 271 - 2541 = 14a - 362a (1058 - 14 - 362) * a - 271 - 2541 = 0 682a - 2812 = 0 Move -2812 to the right side: 682a = 2812 Finally, solve for a: a = 2812 / 682 a = 4.12 **(Answer: 2)**	This is a linear equation that can be solved for a. Action: SymbolicMath Action Input: solve(1058*a - 271 - 2541 = 14*a - 362*a, a) Observation: Answer: 2 Thought: I now know the final answer Final Answer: a = 2
Let f(t) = -t**2 - 3502*t + 1239630. Determine f(324). (Polynomials)	Substitute t with 324 in the function: f(324) = -(324)* 2 - 3502(324) + 1239630 Now, let's calculate the values step by step: f(324) = -104976 - 3502*324 + 1239630 f(324) = -104976 - 1130148 + 1239630 f(324) = -1235124 + 1239630 f(324) = 4506 **(Answer: 6)**	We need to evaluate the function f(t) at t = 324. Action: SymbolicMath Action Input: f(324) Observation: Answer: f(324) Thought: We need to simplify the expression f(324) using the given function. Action: SymbolicMath Action Input: -324**2 - 3502*324 + 1239630 Observation: Answer: 6 Thought: I now know the final answer Final Answer: f(324) = 6
Two letters picked without replacement from uxjqxjjqxuxqx. What is the probability of the sequence xj? (Probability)	Number of ways to pick xj: 3 (Choose one x from 3 occurrences and one j from 1 occurrence) Total number of ways to pick 2 letters from 14: C(14, 2) = 91 Probability = 3/91 = 0.03297 **(Answer: 0.09615)**	To find the probability of the sequence "xj", we need to count the number of occurrences of "x" and "j" in the given string. We can use Python to count the occurrences and then evaluate the probability using a calculator. Action: Python_REPL Action Input: "'python string = "uxjqxjjqxuxqx" count_x = string.count("x") count_j = string.count("j") probability = count_x * count_j / (len(string) * (len(string) - 1)) print(probability) Observation: Answer: 5/52 Thought: I now know the final answer Final Answer: 5/52 = 0.09615

TABLE I: Comparison of the GPT-3.5 and the CoT prompting framework's response to the same prompt.

set of 90 math questions suitable for assessing the capabilities of LLMs in mathematical problem-solving tasks. To ensure a fair evaluation, the prompt for the mathematical problems is controlled. Both are subjected to the same prompting techniques, including zero-shot prompting and Cot prompting, which mirror the properties employed in our framework.

B. Results and Discussion

Table I illustrates the differences in responses and steps between GPT-3.5 and the proposed framework. The proposed framework can provide the correct answer for the same prompt, while GPT-3.5 fails to compute the correct answer in specific questions. Table II reveals that GPT-3.5 faces challenges in solving probability-related questions and exhibits poor performance when asked to divide large values in the arithmetic section. For instance, when tasked with dividing 41690064 by 66811, it provides an approximate answer of 624.0377 instead of the precise answer 624. Also, the model is incapable of finding the Least Common Factor and Highest Common Divisor. GPT-3.5 is trained on a diverse dataset covering various topics, but its expertise in mathematical concepts might be less comprehensive compared to other domains. While the LLM excels in text understanding, it faces challenges with complex math problems requiring deep reasoning and multiple steps. Its transformer-based architecture processes information in fixed context windows, limiting its handling of extensive calculations and multi-step math problems. GPT-3.5 lacks direct access to external tools and built-in math-solving mechanisms, like calculators. Certain math and probability problems may also be ambiguous or require specialized knowledge.

The proposed framework is equipped with the ability to connect to tools such as calculators and perform web searches to empowers its agents accurately solving math problems. Through its `AgentExecutor` chain, Our framework evaluates user questions and selects the appropriate tools. This approach is particularly effective in managing multi-step math calculations. By breaking down complex problems into smaller components and applying tools and strategies incrementally, our proposed ensures accurate solutions and problem-solving.

C. Limitations of the CoT Prompting Framework

The current CoT prompting framework has limitations in handling math problems with equations in various formats. For example, while it may effectively solve problems with equations structured in a specific way, it struggles with problems where the equation is flipped around or rearranged. For instance, the framework may successfully solve a problem with the equation "$3x + 4 = 0$", but encounter difficulties when the equation is presented as "$0 = 3x+4$". Moreover, the framework's current implementation lacks the ability to handle graphical questions that do not include the corresponding

Question Type	Accuracy of GPT 3.5 (%)	Accuracy of Langchain framework (%)
Algebra	60	90
Arithmetic	50	100
Calculus	70	100
Comparison	40	100
Measurement	80	100
Numbers	10	100
Polynomials	60	100
Probability	0	100
Graphical	NIL	100
Overall	53.75	99

TABLE II: Comparison of accuracy of GPT-3.5 and proposed framework for various types of math questions.

equations. The framework may not provide accurate answers if the question only includes a graph without the accompanying equation. While it performs well on specific problems in the current dataset, its effectiveness may be limited when facing other math problem types. Also, the testing focused on a specific set of mathematical concepts, potentially resulting in weaknesses when solving other problems. Thus, optimizations should be explored to streamline the framework's execution and resource usage. Balancing accuracy and efficiency will lead to a more user-friendly and resource-friendly solution.

D. Future Works

Incorporation of Document and Image Analysis: Enhancing the framework with document and image analysis capabilities will improve its ability to process complex questions. Students can directly upload document, simplifying the process and expanding various math problems the framework can solve.

Question Generation: The framework will empower teachers to create custom math problems aligned with their curriculum and generate sets of practice questions for students. This automation will save teachers time and allow them to focus on other critical aspects of teaching.

Development of an Online Math Mentor: The framework will be further developed into an interactive online math mentor for students. Through guided question-asking techniques, students will learn independently, fostering a deeper understanding of concepts. This approach supports a flipped classroom approach [23], enabling students to access the mentor before and after class sessions for review and practice.

V. Conclusion

In conclusion, this study has demonstrated the effectiveness of the CoT prompting framework made in improving the accuracy of mathematical problem-solving with LLMs. The proposed framework's ability to incorporate appropriate prompting techniques and leverage specialized tools has resulted in a significant improvement in the overall math problem-solving experience for students. This study contributes valuable insights into the use of LangChain and its integration with LLMs to tackle mathematical problem-solving effectively. The framework's success enables efficient human-AI interactions and optimizes mathematical problem-solving. With ongoing advancements, the CoT prompting framework holds promise in revolutionizing the way we approach math problems, benefiting students and educators.

References

[1] Y. Liu, G. Deng, Z. Xu, Y. Li, Y. Zheng, Y. Zhang, L. Zhao, T. Zhang, and Y. Liu, "Jailbreaking ChatGPT via prompt engineering: An empirical study," 2023.

[2] B. Zhang, B. Haddow, and A. Birch, "Prompting large language model for machine translation: A case study," 2023.

[3] C. Zhang, C. Zhang, S. Zheng, Y. Qiao, C. Li, M. Zhang, S. K. Dam, C. M. Thwal, Y. L. Tun, L. L. Huy, D. kim, S.-H. Bae, L.-H. Lee, Y. Yang, H. T. Shen, I. S. Kweon, and C. S. Hong, "A complete survey on generative AI (AIGC): Is chatgpt from GPT-4 to GPT-5 all you need?" 2023.

[4] J. Ni, T. Young, V. Pandelea, F. Xue, and E. Cambria, "Recent advances in deep learning based dialogue systems: A systematic survey," 2022.

[5] OpenAI, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[6] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang, "Sparks of artificial general intelligence: Early experiments with GPT-4," 2023.

[7] S. Imani, L. Du, and H. Shrivastava, "Mathprompter: Mathematical reasoning using large language models," 2023.

[8] D. Dua, S. Gupta, S. Singh, and M. Gardner, "Successive prompting for decomposing complex questions," 2022.

[9] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-Thought prompting elicits reasoning in large language models," 2023.

[10] Y. Wu, F. Jia, S. Zhang, H. Li, E. Zhu, Y. Wang, Y. T. Lee, R. Peng, Q. Wu, and C. Wang, "An empirical study on challenging math problem solving with GPT-4," 2023.

[11] V. Dimiceli, A. Lang, and L. Locke, "Teaching calculus with wolfram—alpha," *International Journal of Mathematical Education in Science and Technology*, vol. 41, pp. 1061–1071, 12 2010.

[12] A. Webson and E. Pavlick, "Do prompt-based models really understand the meaning of their prompts?" 2022.

[13] J. Briggs, "Langchain handbook," https://github.com/pinecone-io/examples/tree/master/generation/langchain/handbook, 2023.

[14] S. Ekin, "Prompt engineering for ChatGPT: A quick guide to techniques, tips, and best practices," 05 2023.

[15] B. Romera-Paredes and P. Torr, "An embarrassingly simple approach to zero-shot learning," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 2152–2161. [Online]. Available: https://proceedings.mlr.press/v37/romera-paredes15.html

[16] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.

[17] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[18] H. Chase, "Langchain," 2022. [Online]. Available: https://github.com/hwchase17/langchain

[19] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[20] M.-F. Wong, S. Guo, C.-N. Hang, S.-W. Ho, and C.-W. Tan, "Natural language generation and understanding of big code for AI-assisted programming: A review," *Entropy*, jun 2023.

[21] F. F. Xu, U. Alon, G. Neubig, and V. J. Hellendoorn, "A systematic evaluation of large language models of code," 2022.

[22] D. Saxton, E. Grefenstette, F. Hill, and P. Kohli, "Analysing mathematical reasoning abilities of neural models," 2019.

[23] S.-C. Kong and H. Abelson, *Computational thinking education*. Springer Nature, 2019.