[41] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? *arXiv preprint arXiv:1905.13211*, 2019.

[42] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=5Xc1ecxO1h`.

[43] Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process, 2024. URL `https://arxiv.org/abs/2407.20311`.

[44] Yizhou Zhang, Lun Du, Defu Cao, Qiang Fu, and Yan Liu. An examination on the effectiveness of divide-and-conquer prompting in large language models, 2024. URL `https://arxiv.org/abs/2402.05359`.

[45] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=WZH7099tgfM`.

Appendix

Contents

A Limitations **14**

B Formal Definitions of Simplified Arithmetic Problem **15**

B.1 Problem Formulation 15

B.2 Contrast to vanilla arithmetic problem 16

C Supplementary Details on Real world Experiment for Optimal CoT Length **16**

C.1 Implementation Details 16

C.2 More Experimental Results 17

D Supplementary Details on Real World Experiment for RL Simplicity Bias **19**

E Additional Synthetic Experiment Details **19**

E.1 Training details 19

E.2 Observation of subtask loss 19

F Theoretical Results under Broader Scenarios **19**

F.1 General Errors 19

F.2 Random Error 20

G Proof **20**

G.1 Proof of Proposition 4.2 21

G.2 Proof of Theorem 4.3 21

G.3 Proof of Corollary 4.4 22

G.4 Proof of Theorem F.3 23

G.5 Proof of Theorem F.6 24

G.6 Proof of Corollary 4.5 24

G.7 Technical Lemmas 25

H Pseudo-code of Length-filtered Vote **27**

A Limitations

This proof-of-concept study highlights the critical role of aligning Chain-of-Thought (CoT) length with model capability and task difficulty, particularly in training data. However, accurately estimating the optimal CoT length in real-world scenarios remains challenging due to the complexity and variability of reasoning tasks. While our theoretical and empirical analyses offer practical heuristics for coarse approximation, they may not fully capture the nuances of diverse problem domains or model behaviors. We leave the development of more precise estimation methods and adaptive strategies for optimal CoT length selection in complex, real-world settings as promising directions for future work.

B Formal Definitions of Simplified Arithmetic Problem

To begin, we aim to empirically investigate the relationship between reasoning performance and CoT length. Therefore, we need to control a given model to generate reasoning chains of varying lengths for a specific task. Unfortunately, no existing real-world dataset or model fully meets these strict requirements. Real-world reasoning tasks, such as GSM8K or MATH [9, 18], do not provide multiple solution paths of different lengths, and manually constructing such variations is challenging. Moreover, it is difficult to enforce a real-world model to generate a diverse range of reasoning paths for a given question. Given these limitations, we begin our study with experiments on synthetic datasets.

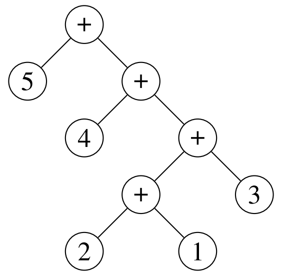

Figure 5: Computation tree of arithmetic expression $5 + (4 + ((2 + 1) + 3))$.

B.1 Problem Formulation

To investigate the effect of CoT length in a controlled manner, we design a synthetic dataset of simplified arithmetic tasks with varying numbers of reasoning steps in the CoT solutions.

Definition B.1 (Problem)**.** In a simplified setting, an arithmetic task q is defined as a binary tree of depth T. The root and all non-leaf nodes are labeled with the $+$ operator, while each leaf node contains a numerical value (mod 10). In addition, we impose a constraint that every non-leaf node must have at least one numerical leaf as a child.

The bidirectional conversion method between arithmetic expressions and computation trees is as follows: *keeping the left-to-right order of numbers unchanged, the computation order of each "+" or tree node is represented by tree structure or bracket structures.* For example, consider the task $5 + (4 + ((2 + 1) + 3))$ with $T = 4$. The corresponding computation tree is defined as Figure 5.

To ensure that CoT solutions of the same length have equal difficulty for a specific problem, we assume that each reasoning step performs the same operations within a single CoT process.

Definition B.2 (Solution)**.** We define a t-hop CoT with a fixed each step length of t as a process that executes t operations starting from the deepest level and moving upward recursively.

According to this definition, the execution sequence is uniquely determined. For example, one way to solve expression in Figure 5 is by performing one addition at a time:

$$5 + (4 + ((2 + 1) + 3)) = \texttt{<1>} \tag{3}$$
$$2 + 1 = 3 \tag{4}$$
$$3 + 3 = 6$$
$$4 + 6 = 0$$
$$5 + 0 = 5\texttt{<END>}.$$

Another approach is to perform two additions at a time:

$$5 + (4 + ((2 + 1) + 3)) = \texttt{<2>} \tag{5}$$
$$(2 + 1) + 3 = 6$$
$$5 + (4 + 6) = 5\texttt{<END>}.$$

The latter approach is half as long as the former, but each reasoning step is more complex[6]. This illustrates a clear trade-off between the difficulty of each subtask and the total number of reasoning steps.

[6]This is because performing two operations at once requires the model to either memorize all combinations of numbers in a two-operator equation and their answers, apply techniques like commutativity to reduce memory requirements, or use its mental reasoning abilities to perform the two operations without relying on CoT.