Solution Path Routing for Accurate and Efficient Language Model Reasoning

Anonymous ACL submission

Abstract

Language models can solve problems in multiple ways. For instance, they can reason step-by-step in natural language, or generate a program that can produce the final answer. In this work, we first empirically demonstrate that there is no one-size-fits-all solution; in some cases code is a better option with respect to accuracy and token-efficiency, but in other cases only natural language allows a correct answer to be found. We then examine language models' ability to appropriately perform *solution path routing*, choosing the most appropriate solution path based on the problem. We find that models struggle to pick the most appropriate solution path simultaneously with solving the problem, but by using a 2-stage pipeline with explicit routing and then problem solving we are able to achieve efficiency gains and sometimes performance improvements.

1 Introduction

Chain of thought reasoning in natural language (Wei et al., 2023; Kojima et al., 2023) is a staple of language model inference today, but it is not universally beneficial (Sprague et al., 2024). Alternatively, language models may be prompted to generate programs that are executed to achieve a solution, a method that is particularly effective in tasks that involve numerical reasoning (Gao et al., 2023; Chen et al., 2023; Bi et al., 2023; Li et al., 2024). Contemporary works have further combined natural language and programmatic solutions either as fall back (Liu et al., 2023; Xiang Yue, 2023), an integrated snippet within the response (Wang et al., 2023; Wen et al., 2024), or in parallel with an answer finalization stage (Hu et al., 2023; Zhao et al., 2023; Xiong et al., 2024). These methods can increase performance, but at a greater cost than simply using natural language or programs due to the need to perform multiple inferences.

In this work, we examine the possibility of *solution path routing*, where we decide, *a priori*, whether to solve a problem using natural language or programming language. Similar to work on model routing (which routes between different LLMs; Shen et al. (2023)), this will make it possible to achieve both accuracy and efficiency improvements, by choosing the best model. We find that language models struggle to simultaneously route to the appropriate solution path and solve the problem. By using a 2-stage pipeline (Figure 1) with explicit routing and then problem solving, we are able to address this routing problem and achieve efficiency gains with competitive performance over exclusively natural language solutions.

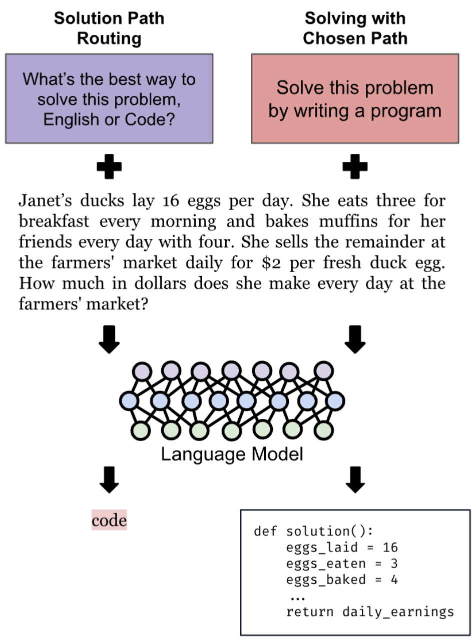

Figure 1: Two step pipeline to routing a model towards a solution method. The model is prompted to first decide whether a problem should be solved with either natural language (NL) or programming language (PL). The output of this step is used to determined which prompt should be used that would direct the model towards the chosen method.

Our findings show that with routing, models

Task	Model (Instruct)	Avg Correct Answer (% Instances)				Avg Tokens (# Tokens)	
		Neither	NL	PL	PL+NL	NL	PL
GSM-Hard	Llama-3.1-8B	34.34	4.93	33.81	26.91	304.89	138.27
	Llama-3.1-70B	31.92	2.96	22.97	42.15	264.25	86.30
	Qwen-2.5-7B	30.25	3.64	22.82	43.29	324.24	108.68
	Qwen-2.5-14B	30.86	2.65	17.44	49.05	341.57	124.36
	Qwen-2.5-32B	33.36	2.88	42.08	21.68	325.94	112.00
	Qwen-2.5-72B	25.47	2.35	20.09	52.08	354.56	97.92
	Task Avg	31.03	3.23	26.54	39.20	319.24	111.26
MathQA	Llama-3.1-8B	34.97	16.52	11.79	36.72	346.82	153.81
	Llama-3.1-70B	23.82	12.80	6.80	56.58	305.99	124.90
	Qwen-2.5-7B	30.52	7.27	28.01	34.20	385.13	148.67
	Qwen-2.5-14B	24.25	10.52	14.44	50.79	391.62	146.86
	Qwen-2.5-32B	28.94	9.51	24.49	37.05	372.32	141.21
	Qwen-2.5-72B	27.20	4.49	22.95	45.36	418.10	132.57
	Task Avg	28.29	10.18	18.08	43.45	370.00	141.34
FinQA	Llama-3.1-8B	48.82	5.75	29.90	15.52	234.60	174.02
	Llama-3.1-70B	35.40	2.88	33.39	28.33	182.63	87.63
	Qwen-2.5-7B	38.54	4.36	30.34	26.77	292.58	124.26
	Qwen-2.5-14B	33.74	3.31	37.23	25.72	293.38	111.51
	Qwen-2.5-32B	32.96	0.70	62.77	3.57	278.05	102.21
	Qwen-2.5-72B	30.43	3.23	32.17	34.18	319.90	101.90
	Task Avg	36.65	3.37	37.63	22.35	266.86	116.92
TabMWP	Llama-3.1-8B	32.06	5.95	27.28	34.71	205.73	148.84
	Llama-3.1-70B	17.17	4.91	28.04	49.88	168.38	99.15
	Qwen-2.5-7B	16.50	5.37	32.29	45.84	206.92	105.35
	Qwen-2.5-14B	16.51	3.03	30.04	50.42	223.62	95.59
	Qwen-2.5-32B	15.18	1.03	44.51	39.28	223.55	112.59
	Qwen-2.5-72B	14.56	1.08	28.69	55.67	237.84	87.33
	Task Avg	18.66	3.56	31.81	45.97	211.01	108.14

Table 1: Average performance for each model based on how much neither PL and NL solutions were right, either NL or PL solutions are right or both were right for every instance in the task.

can perform well on mathematical reasoning tasks where it may not necessarily be clear whether a problem should be solved by programming language or natural language.

2 The Importance of Considering Different Solution Paths

First, to demonstrate the value of considering different solution paths, we prompt the instruction-tuned version of the Llama-3.1 (Grattafiori et al., 2024) and Qwen-2.5 (Team, 2024) models with two different solutions paths in a 0-shot setting; programming language (PL prompt style) to produce executable programs that return the answer and natural language (NL prompt style) to write step-by-step rationalization before arriving at an answer. Both prompts can be found in Appendix B. We evaluate the models on mathematical reasoning tasks that describe mathematical problem in natural language such as GSM Hard (Gao et al., 2023) and MathQA (Amini et al., 2019), and reasoning tasks over tabular information such as FinQA (Chen et al., 2022), and TabMWP (Lu et al., 2023).

In Table 1. We measure the average number of instances where using neither NL or PL prompts resulted in the model answering correctly, either NL or PL are exclusively correct, or both the NL and PL paths arrived at the correct answer. Compared

Model (Instruct)	Commonsense QA (% NL Response)		GSM8k (%PL Response)	
	Direct	Select	Direct	Select
Llama-3.1-8B	32.72	84.89	27.90	62.47
Llama-3.1-70B	100.00	99.47	0.27	25.40
Qwen-2.5-7B	100.00	99.67	2.16	82.90
Qwen-2.5-14B	100.00	99.55	0.04	18.12
Qwen-2.5-32B	100.00	99.59	0.08	15.01
Qwen-2.5-72B	100.00	99.75	0.53	62.93
Model Avg	83.18	95.91	7.39	44.34

Table 2: Proportion of responses in NL for CommonsenseQA and PL in GSM8k for models prompted to (a) provide a solution based on described choices of PL or NL (*Direct*) or (b) provide a response between "programming language" or "natural language" as the best solution to solve a given problem (*Select*).

to using NL, there are substantially more instances (which can be as high as 11.16x as much in FinQA) where using PL resulted in the right answer. However, we also see that there are in fact instances where using PL does not arrive at the right answer and using NL instead does. Additionally we calculate the average response length when solving with NL or PL and see that PL always results in less tokens being used. This suggests that being able to route towards one or the other would result in performance with higher accuracy overall while also maintaining efficiency over just using natural language.

3 Solution Routing

3.1 Models Cannot Route Consistently

We test how well various language models can simultaneously route and solve a given problem. For a given instance, a language model is prompted to solve with either thinking step-by-step (natural language) or writing a program (programming language). Both options are given details such as the format of the final answer or what the program method name should be for post-processing (full list of prompts in Appendix-B). were tested on on GSM8k (Cobbe et al., 2021) and CommonsenseQA (Talmor et al., 2019). Both being distinct from each other with the former likely benefiting from programmatic solutions more than the latter.

In Table 2, the models we tested were able to consistently solve CommonsenseQA with natural language but struggle to accurately resolve GSM8k with programming language and instead gravitate towards natural language solutions as show by the

Model (Instruct)	Solve with	MATH %Acc	MATH #Tokens	MATH %PL	MATH %Code	MathQA %Acc	MathQA #Tokens	MathQA %PL	MathQA %Code
Llama-3.1-8B	PL	23.59	**344.00**	100.00	83.62	48.51	**153.00**	100.00	96.85
	NL	19.91	1225.00	0.00	0.00	**53.23**	346.00	0.00	0.00
	Routing	**24.28**	770.00	51.01	84.57	51.12	245.00	51.73	96.82
Llama-3.1-70B	PL	**40.97**	**218.00**	100.00	96.21	63.38	**124.00**	100.00	95.89
	NL	38.11	1108.00	0.00	0.00	**69.38**	305.00	0.00	0.00
	Routing	40.04	598.00	62.04	95.60	65.06	190.00	63.65	95.87
Qwen-2.5-7B	PL	**38.48**	**274.00**	100.00	91.07	**62.21**	**148.00**	100.00	95.57
	NL	28.46	802.00	0.00	0.00	41.47	385.00	0.00	0.00
	Routing	36.21	620.00	41.31	71.99	58.16	194.00	80.97	95.46
Qwen-2.5-14B	PL	39.06	**196.00**	100.00	93.97	**65.23**	**146.00**	100.00	94.51
	NL	37.33	867.00	0.00	0.00	61.31	391.00	0.00	0.00
	Routing	**42.10**	593.00	37.32	88.85	62.48	341.00	20.50	94.19
Qwen-2.5-32B	PL	**52.69**	**454.00**	100.00	52.41	**61.54**	**141.00**	100.00	92.98
	NL	52.64	690.00	0.00	0.00	46.57	372.00	0.00	0.00
	Routing	51.75	660.00	54.27	26.80	51.42	298.00	31.83	92.99
Qwen-2.5-72B	PL	47.04	**267.00**	100.00	79.82	**68.31**	**132.00**	100.00	93.86
	NL	39.29	879.00	0.00	0.00	49.85	418.00	0.00	0.00
	Routing	**49.57**	575.00	47.06	83.96	64.12	194.00	77.79	93.87

Table 3: Aggregate performance for MATH and MathQA. Models are measured by Accuracy (**%Acc**) and Efficiency (response length denoted by **#Token**). In addition, we track **%PL** that shows how many instances were routed towards the programming language solution path and **%Code** that refers to the average proportion of a PL-routed response being actually code (Responses can start with *"here is the code to solve"*). **Bold** numbers refer to best among the 3 solution strategy (higher is better for %Acc, lower is better for #Tokens) while underlined numbers show the second best.

percentage of responses in PL being 2% or less except for Llama-3.1-8B (*Direct* in Table 2). For the case of Llama-3.1-8B, we observe only 32.72% of problems solved with natural language which conversely means that 67.28% of the responses where in programming. We find that qualitatively these answers are not informative. Often in the form of `"def solution():\n\treturn 'A'"` which we deem as an uninformative solution.

Alternatively, we try the *Select* prompt to decide the best way to solve a given problem with the choices being "programming language" or "natural language". Models are able to substantially increase the routing rate to use programming language for GSM8k from on average 7.39% to 44.34% while maintaining a high selection rate for natural language on CommonsenseQA.

3.2 2-Stage Routing

As Table 2 suggests, models struggle to route towards programming language solutions directly but can route at a higher success rate when only asked to answer which solution is the best for a given problem. To enable language models to better route to a solution path, we propose a 2-stage pipeline that consists of the *Routing* stage that utilizes the *Select* prompt to choose a solution path and the *Solving* stage that proceeds to prompt the model to solve an instance with a particular method. For example, as illustrated in Figure 1, after the first stage where the model answers "programming language", the model is then prompted to implement a programmatic solution.

We evaluate this method on the test split of MATH (Hendrycks et al., 2021) (Level 5 problems) and MathQA to highlight how this method can be used to solve challenging mathematical reasoning tasks. Our motivation to use MATH and MathQA stems their popularity in demonstrating the natural language reasoning capabilities in language models. The MATH benchmark also features 7 categories that can provide insight to how well routing between NL and PL can be for each different types of math problems. We use sampling parameters Temperature of 0.6 and Top-p of 0.9 for all evaluations.

Experiment results (Table 3) indicate that in 11 out of 12 cases, solution path routing achieves either the best or second-best accuracy, indicating that it stably strikes a good balance between NL and PL-based solutions, making it a safe choice to use in situations where it is not clear a-priori which of the two methods is appropriate. Further, compared to solely using *NL*, it is more efficient,

Model (Instruct)	Solve with	Prealgebra		Algebra		Number Theory		Counting & Probability		Geometry		Intermediate Algebra		Precalculus	
		%PL	%Code	%PL	%Code	%PL	%Code	%PL	%Code	%PL	%Code	%PL	%Code	%PL	%Code
Llama-3.1-8B	PL	100.00	85.36	100.00	82.78	100.00	81.96	100.00	72.18	100.00	90.34	100.00	87.26	100.00	85.46
	Routing	47.15	84.31	52.77	82.65	66.23	86.49	76.42	70.80	53.03	89.81	31.07	90.92	30.37	86.97
Llama-3.1-70B	PL	100.00	95.72	100.00	96.22	100.00	95.12	100.00	96.70	100.00	97.15	100.00	95.40	100.00	97.12
	Routing	59.59	96.11	65.47	96.80	46.10	93.91	61.79	94.13	68.18	97.38	64.29	94.96	68.89	95.89
Qwen-2.5-7B	PL	100.00	93.69	100.00	89.74	100.00	95.20	100.00	93.31	100.00	86.63	100.00	90.71	100.00	88.23
	Routing	40.41	80.97	47.56	93.98	59.09	72.83	49.59	94.80	16.67	94.73	42.50	37.16	33.33	29.48
Qwen-2.5-14B	PL	100.00	95.15	100.00	94.38	100.00	93.96	100.00	90.00	100.00	95.76	100.00	93.52	100.00	95.02
	Routing	17.62	94.04	34.85	87.20	61.69	90.48	50.41	85.60	12.12	84.26	46.79	84.35	37.78	96.05
Qwen-2.5-32B	PL	100.00	72.19	100.00	48.74	100.00	62.18	100.00	70.75	100.00	44.94	100.00	34.50	100.00	33.59
	Routing	29.53	40.16	51.47	24.70	76.62	34.22	62.60	34.71	21.97	16.81	71.79	19.04	65.93	17.95
Qwen-2.5-72B	PL	100.00	83.62	100.00	78.40	100.00	82.20	100.00	81.04	100.00	77.91	100.00	78.32	100.00	77.30
	Routing	52.33	88.94	48.86	92.80	83.77	80.52	43.90	93.37	31.06	75.29	33.93	80.01	35.56	76.82

Table 4: For each category in MATH, the number of instances that were routed to PL and percentage of the content in among them that was code.

decreasing tokens generated from an average of 928 to 636 tokens on MATH, and 370 to 244 tokens on MathQA.

Overall, the model-routing process chooses to solve the problem in PL on average around 50% of the time. Models also exhibit high ratios of code content in the responses that were routed to PL with the exception of 1 model. We attribute the portion of responses not being code to a model's tendency to start with natural language such as *"To solve this..."*. One outlier was Qwen-2.5-32B, which only had 26.80% of its average response being detected as code despite similar rates of routing with other models, which could indicate this particular model gravitates towards natural language. Qualitatively, we find that in these cases, models tend to explain their code which leads to low rates of code content.

We further break down MATH into its subcategories in Table 4 to observe how well models route under different mathematical problems in the benchmark. Interestingly, we do not see consistency between the rate of which models are routed towards PL and any particular MATH category. Some models may have high routing rates for Counting and probability while others on Number Theory. This suggests that while models are versatile in writing programs to solve a given problem, they may not be adequately trained to identify which types of problems may be better solved with certain solution paths in a way that balances accuracy and efficiency.

4 Related Work

Previous works have explored model inference that generate both PL and NL solutions simultaneously (Zhao et al., 2023). After generation, the models are prompted to reflect on these generations and choose which one to commit to based on how the model perceives it's correctness. Similarly, Hu et al. (2023) generates both solution path and combines them for an additional round of step-by-step reasoning to arrive at the final answer.

In contemporaneous work, Xiong et al. (2024) studied how models may solve mathematical tasks by choosing from multiple methods. Our works differ in scope whereas ours study the ability of open models at various sizes and routing capability measured by the rate the answers are code which inspired our proposed 2-stage solution path routing. We also identify improvements of both accuracy and efficiency.

5 Conclusion

We show the benefits routing through by comparing performance and efficiency between NL and PL solution prompts. We then show how models struggle to consistently route between the two choices. Finally, using a 2-stage solution path routing, models are able to select between distinct solution implementations that enables them to perform as well or better and more efficiently compared to only utilizing natural language reasoning responses. We find that this method is effective in steering models away from gravitating towards natural language responses.

Limitations

Our work is limited to evaluating open and instruction-tuned models in English. The scope of this work does not consider closed API-form models. Additionally our work only considers prompt-base evaluation and does not analyze nor evaluate the impacts of supervised finetuning or other post-training techniques.

As model responses can be code, we require the generated code to be executed to derive the final answer. As such, there may be risks where the model generates harmful and/or malicious code. When similar methods are used in actual deployed systems, reasonable precautions must be taken for security sandboxing.

References

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *Preprint*, arXiv:1905.13319.

Zhen Bi, Ningyu Zhang, Yinuo Jiang, Shumin Deng, Guozhou Zheng, and Huajun Chen. 2023. When Do Program-of-Thoughts Work for Reasoning? *arXiv preprint*. ArXiv:2308.15452.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. ArXiv:2211.12588.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2022. Finqa: A dataset of numerical reasoning over financial data. *Preprint*, arXiv:2109.00122.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided Language Models. *arXiv preprint*. ArXiv:2211.10435 [cs].

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Yi Hu, Haotong Yang, Zhouchen Lin, and Muhan Zhang. 2023. Code prompting: a neural symbolic method for complex reasoning in large language models. *Preprint*, arXiv:2305.18507.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large Language Models are Zero-Shot Reasoners. *arXiv preprint*. ArXiv:2205.11916.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Chengshu Li, Jacky Liang, Andy Zeng, Xinyun Chen, Karol Hausman, Dorsa Sadigh, Sergey Levine, Li Fei-Fei, Fei Xia, and Brian Ichter. 2024. Chain of code: Reasoning with a language model-augmented code emulator. *Preprint*, arXiv:2312.04474.

Tengxiao Liu, Qipeng Guo, Yuqing Yang, Xiangkun Hu, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2023. Plan, verify and switch: Integrated reasoning with diverse x-of-thoughts. *Preprint*, arXiv:2310.14628.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2023. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *International Conference on Learning Representations (ICLR)*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugging-gpt: Solving ai tasks with chatgpt and its friends in hugging face. *Preprint*, arXiv:2303.17580.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2024. To CoT or not to CoT? Chain-of-thought helps mainly on math and symbolic reasoning. ArXiv:2409.12183 [cs].

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2023. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. *Preprint*, arXiv:2310.03731.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Jiaxin Wen, Jian Guan, Hongning Wang, Wei Wu, and Minlie Huang. 2024. Unlocking reasoning potential in large langauge models by scaling code-form planning. *Preprint*, arXiv:2409.12452.

Ge Zhang Yao Fu Wenhao Huang Huan Sun Yu Su Wenhu Chen Xiang Yue, Xingwei Qu. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Xuyuan Xiong, Simeng Han, Ziyue Zhou, and Arman Cohan. 2024. Inc-math: Integrating natural language and code for enhanced mathematical reasoning in large language models. *Preprint*, arXiv:2409.19381.

James Zhao, Yuxi Xie, Kenji Kawaguchi, Junxian He, and Michael Xie. 2023. Automatic Model Selection with Large Language Models for Reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 758–783, Singapore. Association for Computational Linguistics.

A Compute Used

This work was done on instances of NVIDIA A6000s and L40s GPU accelerators. For each Model-Prompt-Benchmark, runtime varied between 10 minutes to 3 hours depending on the size of the models. We utilize the VLLM (Kwon et al., 2023) library with pipeline parallelization to run multiple parallel inference process. For models of 32 billion parameters and above, we additionally utilize tensor-parallel to split the models parameters.

B System Prompts

We list the prompts we used in to direct models for every problem instance it sees in Table 5. Results in Table 2 are average across variations i.e results from *Direct* (a) and (b) and *Select* (a) and (b) are averaged.

Prompt Style	Prompt Text
PL	`Solve the following problem by DIRECTLY and ONLY writing a PYTHON program. The answer must be a function named solution() without any input arguments. The function MUST return an single value.`
NL	`Solve the following problem by thinking step-by-step. Derive and go through the logical steps in order to arrive at the final answer. At the end, you MUST write the answer after 'The answer is'.`
Direct (a)	`Choose only one way to solve the problem: by thinking step-by-step OR writing a program as a way to solve a given task. Do NOT use both:\n1. Thinking step-by-step: Think step-by-step. Derive and go through the logical steps in order to arrive at the final answer. At the end, you MUST write the answer after 'The answer is'.\n2. Writing a program: DIRECTLY and ONLY write a program with the PYTHON programming language. The function must be named solution() without any input arguments.\nAt the end, you MUST return an single value.`
Direct (b)	`Choose only one way to solve the problem: by writing a program OR thinking step-by-step as a way to solve a given task. Do NOT use both:\n1. Writing a program: DIRECTLY and ONLY write a program with the PYTHON programming language. The function must be named solution() without any input arguments. At the end, you MUST return an single value.\n2. Thinking step-by-step: Think step-by-step. Derive and go through the logical steps in order to arrive at the final answer.\nAt the end, you MUST write the answer after 'The answer is'.`
Select (a)	`Based on a given task, choose only one way that can be used to solve the problem: by natural language OR programming language to solve a given task. Do NOT use both. Answer with either "natural language" or "programming language".`
Select (b)	`Based on a given task, choose only one way that can be used to solve the problem: by programming language OR natural language to solve a given task. Do NOT use both. Answer with either "programming language" or "natural language".`

Table 5: Prompts used in this work. We alter the prompts for *Direct* and *Select* by switching the order of solution path choices.