Less is More: Compressed Reasoning with Large Language Models via Structured Prompting

Anonymous ACL submission

Abstract

Recent breakthroughs in LLMs have significantly enhanced their abilities to reason and solve thinking problems in various domains. Reinforcement learning (RL) and Supervised Fine Tuning (SFT)-based post-training mechanisms along with high-quality curated data have enabled models such as DeepSeek-R1, Qwen2.5-Math, OpenAI o1 etc to outperform the state-of-the-art even in challenging benchmarks such as AIME'24 and MATH-500. However, a significant drawback of these models is the large Chain-of-Thought (CoT) generation step required to get to the final response, increasing resource requirement and response time. RL-based approaches with rewards for brevity as well as accuracy reduce verbosity, but require custom training with multiple generations per problem, involving significant resource usage, often limiting practitioners to small LLMs. Additionally, the performance lift obtained can be inconsistent. In this paper, we introduce TeleMathLang, a minimal syntax for reasoning and math that enables LLMs to generate complete chains of reasoning while reducing response length by 30-65% across GSM8K, AI2-ARC, and MATH-500. We show that LLMs condense their responses when TeleMathLang is used purely as a prompting strategy as well as for finetuning (even small LLMs with 1.5B parameters). Further, we show that it outperforms other concise reasoning prompts in accuracy as well as semantic entropy, preserving what makes CoT work while reducing verbosity.

1 Introduction

Recent advances in large language models (LLMs) have led to remarkable improvements in the abilities of LLMs to perform complex reasoning and mathematical tasks, with models such as DeepSeek-R1, Qwen 2.5 Math and OpenAI o1 (DeepSeek-AI et al., 2025; OpenAI et al., 2024; Qwen et al., 2025) showing impressive results on challenging math and reasoning benchmarks such as AI2-ARC, AIME'24, MATH-500 (Clark et al., 2018; HuggingFaceH4, 2024; Hendrycks et al., 2021) etc. This can largely be attributed to extensive post-training that aims to incentivise the generation of large chains-of-thought (CoT) (Wei et al., 2023) before returning the final output. This phenomenon in three recent works is illustrated in Figure 1. However, while accurate, CoT-based responses are often highly verbose with lengthy explanations which do not always help performance (Aggarwal and Welleck, 2025; Fatemi et al., 2025), inflating inference cost and latency. At the same time, attempting to reduce the CoT output length through specifying number of reasoning steps or output tokens leads to degraded performance (Jin et al., 2024).

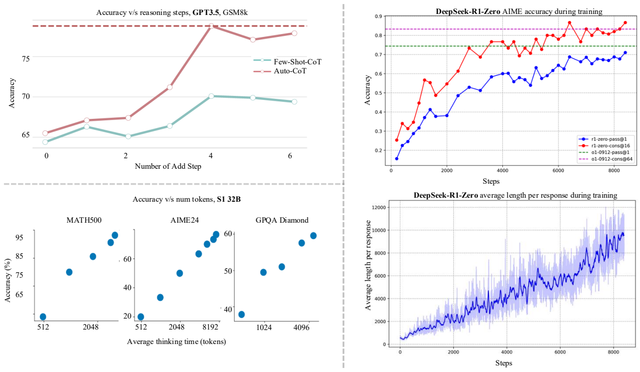

Figure 1: Correlation between large response size and accuracy in recent works. Top left from Jin et al. (2024) bottom left from (Muennighoff et al., 2025), right from DeepSeek-AI et al. (2025)

One line of prior work has explored reinforcement learning (RL) techniques to shorten reasoning chains while preserving accuracy. For example, specialized RL algorithms such as Group Relative Policy Optimization (GRPO) (DeepSeek-AI et al., 2025) have been applied with reward functions that encourage shorter correct answers and penalize overly long solutions (Fatemi et al., 2025; Aggarwal and Welleck, 2025; Liu et al., 2025; Hou et al., 2025). In practice, these RL-based methods can compress chains-of-thought. However, RL approaches tend to be computationally expensive, often requiring iterative training phases (Hou et al., 2025) and heavy compute investment - for instance, GRPO on a 1.5B parameter LLM with 6 generations with only 4096 tokens per step can require as many as 4 48GB GPUs to complete 1 epoch in 1 day on 40k training samples (Dang and Ngo, 2025). This practically limits a large number of users to small, open-source models. Moreover, fine-tuning LLMs with RL can be unstable and brittle – small changes in hyperparameters or even random seeds can lead to large variance in results (Hochlehnert et al., 2025). Finally, aiming to reduce output length through reward functions can lead to LLMs learning reasoning processes through reward hacking (Gao et al., 2022).

In this paper, we introduce TeleMathLang, a simple, minimal syntax comprising of reasoning tokens and rules, that generates token-efficient reasoning chains without sacrificing accuracy. TeleMathLang works with instruction-tuned models by passing the syntax via prompts, and with heavily post-trained models (that work best under certain system and user prompts such as Qwen et al. (2025)), we show that finetuning on small datasets with TeleMathLang-formatted solutions for few epochs is sufficient to reduce token count without sacrificing accuracy (section 4.1.4). Under both these setups, it is resource efficient unlike RL-based methods. Further, we show that it outperforms other concise prompting techniques (Lee et al., 2025; Xu et al., 2025), without leading to increased generation uncertainty, which can be quantified by measuring semantic entropy (Shannon, 1948; Farquhar et al., 2024) (section 4.1.3).

To summarize, we make the following contributions:

- We introduce TeleMathLang: a novel reasoning syntax comprising of thinking tokens and rules which enables LLMs to generate complete reasoning chains, achieving CoT-level accuracy on complex reasoning tasks while reducing token count significantly.
- We apply TeleMathLang to public math and reasoning benchmark datasets (MATH-500, GSM8K and AI2-ARC) with 5 different LLMs under prompt-based and and finetuned setups, and show that our method consistently achieves comparable or higher accuracy compared to CoT with token count reduction up to 65%+, outperforming other concise prompting strategies.
- Using the concept of semantic entropy, we show that TeleMathLang does not increase response uncertainty relative to CoT reasoning chains even in complex problems, unlike other concise reasoning methods. We also show that it is able to automatically adapt reasoning length to problem complexity, ensuring token efficiency does not sacrifice accuracy.

2 Related Work

LLM Reasoning Brown et.al. (Brown et al., 2020) showed that LLMs are able to generalize to unseen tasks if few illustrative examples were included in the prompt. Chain-of-Thought (CoT) prompting (Wei et al., 2023) significantly improved LLM performance in tasks that required reasoning, through the usage of few-shot examples that showed multi-step reasoning. By breaking down complex problems into smaller, manageable steps, CoT allows LLMs to generate chains of reason that lead to the correct answer (discussed in more detail section 3). Kojima et al. (Kojima et al., 2023) showed that models can be encouraged to generate chains of thoughts even without few-shot examples. More sophisticated structured reasoning methods have been proposed more recently (Yao et al., 2023a; Chen et al., 2023; Xu et al., 2025; Yao et al., 2023b), which attempt to increase accuracy by maintaining multiple reasoning chains, using computational tools, or multi-step reason-observe-act chains. However, CoT remains widely used.

More recently, thinking models have been developed that are explicitly designed to think longer during inference (DeepSeek-AI et al., 2025; Qwen et al., 2025; Shao et al., 2024; OpenAI et al., 2024). These models are trained on high-quality supervised fine-tuning (SFT) data with responses in CoT and program-of-thought (PoT) format (Chen et al., 2023). SFT checkpoints are then further incentivized to generate larger CoT through reinforcement learning (RL). DeepSeek R1 (DeepSeek-AI et al., 2025) reported that during the course of extensive reinforcement learning, the model learned to correct itself and generate longer chains. These recent reasoning models have achieved state-of-the-art performance on LLM benchmarks across domains. However, CoT-based training often biases models to generate responses with high verbosity and lengthy explanations which do not always help performance (Aggarwal and Welleck, 2025; Fatemi et al., 2025), inflating inference cost and latency. It can be shown that RL objectives can bias models to generate long chains if intermediate steps are suboptimal (Liu et al., 2025), and it has been noticed that comparatively larger chains are often associated with incorrect responses (Aggarwal and Welleck, 2025).

Concise Reasoning To address the verbosity in CoT responses, studies have explored concise prompting strategies. Lee et al. (2025) explore the performance of a host of prompting strategies across multiple LLMs. (Xu et al., 2025) propose chain-of-draft, where LLMs are instructed to limit each thinking token to 5 words at most. Kang et al. (2024) proposes finetuning on a small training dataset with short-form CoT solutions (C3oT). While these approaches are promising, they have limitations. Chain-of-draft shows noticeably poor performance in small LLMs, while C3oT requires finetuning and will not work with proprietary LLMs, making both these approaches less generalizable than ours. Additionally, we discuss theoretically why concise prompting in suboptimal in section 3, and in section 4, we show the drawbacks of concise prompting with respect to both accuracy and uncertainty (measured with semantic entropy).

Several studies have also explored reinforcement learning (RL) approaches aimed at producing more concise reasoning paths. Liu et al. (2025) propose a modification to GRPO by removing normalization term in the objective. Aggarwal and Welleck (2025); Fatemi et al. (2025) propose modifications to GRPO using token constraints, and Hou et al. (2025) uses an iterative approach to reduce token count gradually. While these approaches show promise, they often require substantial computational resources and can be unstable during training. Additionally, the gains made by RL-based methods are brittle, as shown by (Hochlehnert et al., 2025). In this paper, we will show that TeleMathLang is able to generate concise reasoning without significant computational overhead consistently across LLMs and datasets.

Semantic entropy Shannon (Shannon, 1948) introduced the concept of entropy in information theory to quantify the amount of uncertainty or randomness associated with a source of information. In NLP, entropy is associated with the probability distribution of tokens. In the presence of uncertainty, none of the probabilities is particularly high. However, in natural language responses difference of tokens is less relevant than the difference in semantic meaning. For example, "Best of Luck!" and "Break a Leg!" use completely different tokens but are semantically equivalent. Farquhar et al. (2024) introduced the concept of semantic entropy to address this. However, this is notoriously difficult to estimate due to the vagueness of semantic equivalence. Methods such as Semantic Entity Probes (Kossen et al., 2024) have been proposed to approximate semantic entropy scalably. In this paper, we introduce a simple pairwise distance-based metric for semantic entropy (section 4.1.3).

3 Proposed Methodology

In this section, we first provide a theoretical analysis of what causes CoT to be so effective in boosting an LLM's ability to reason using the vocabulary of Boolean circuit complexity. We then discuss how our proposed method, TeleMathLang, aims to preserve CoT performance boost while avoiding the drawbacks of concise prompting instructions.

3.1 Preliminaries

Boolean Circuit Complexity Boolean circuits provide a useful framework for understanding the computational limits of transformer architectures. A Boolean circuit, formally, is a directed acyclic graph where nodes are AND, OR, or NOT gates. A *circuit* consists of input and output layers, with feedforward connections (composed of logic gates) between each other. The circuit's **depth** is the longest path from input to output, while the **size** of the circuit is the total number of gates in it (Li et al., 2024). Boolean circuit complexity classes categorize computational problems based on the shape and depth of Boolean circuits (networks of AND/OR/NOT gates) needed to solve them. AC^0 and TC^0 are fundamental classes in this hierarchy, comprising circuits are highly parallel but shallow, able to perform computations that do not involve a large number of sequential steps. Definitions of these classes can be found in appendix A.

Transformer expressivity Transformers (Vaswani et al., 2017) are designed to allow parallel training, capturing positional information through positional encodings, instead of sequential training as in RNNs. Through the self-attention mechanism, transformers learn the dependencies between all input tokens, eliminating issues such as failure to learn long term dependence. Despite their impressive capabilities, it has been shown that transformer expressivity can be bounded by low-complexity classes such as TC^0 with log precision assumption (Merrill and Sabharwal, 2023) and AC^0 with constant-bit precision (Li et al., 2024), as they are not designed to perform deep sequential computations unlike RNNs.

3.2 Why does CoT work so well?

CoT enhances transformer expressivity In CoT prompting, each reasoning step (each intermediate token or sentence the model generates) can be viewed as a new input for the next step. The transformer processes the input (which now includes the previously generated thought) with its fixed layers, produces the next step of the chain-of-thought, essentially turning a single deep computation into what Li et al. (2024) calls "a sequence of shallow computations". Figure 2 illustrates this with an example.

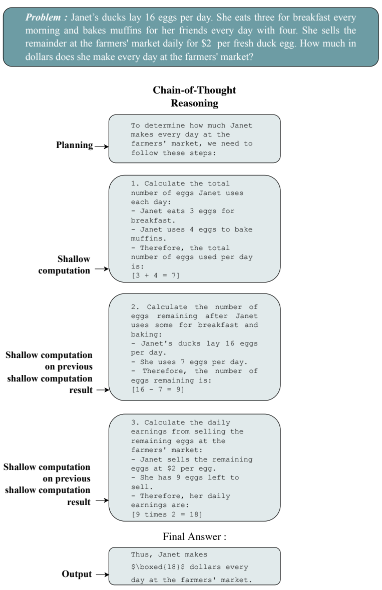

Figure 2: CoT reduces complex operations to a sequence of simpler sequential operations, including previous steps in context

Li et al. (2024) prove that with T steps of CoT, a constant-depth transformer (even with constant-bit precision and modest width like O(log n)) can simulate any Boolean circuit of size T. A number of steps polynomial in the input length in theory allows a transformer to solve problems in P (polynomial time), because poly-size circuits can be simulated by sufficient CoT steps.

This theoretical result is confirmed by empirical works such as Madaan et al. (2023) which shows that the form of CoT is more important that the content. However, at the same time, it is crucial to make each CoT step informative, solving problems with circuit complexity AC^0, to ensure the increased output length actually increases expressivity. This has been shown for example in ablations in (Wei et al., 2023; Lanham et al., 2023), where simply increasing inference-time token count does not work. Any reasoning vocabulary proposed as an alternative to CoT (which consists typically of a few demonstrations or a general instruction to start with "Let's think step-by-step") should have sufficient vocabulary to handle problems with this complexity.

Finally, with works such as DeepSeek-AI et al. (2025), recently LLMs are tuned on large corpuses with CoT outputs included in the response. This further helps boost CoT performance during inference.

Limitations of concise prompting instructions Works such as Lee et al. (2025); Kang et al. (2024) explore prompting instructions and fine-tuning strategies to reduce CoT length in LLM responses. However, these instructions miss out on the key insight that CoT allows transformers to be more expressive by enabling them to perform serial computations. General instructions to "Be Concise" or to not use proper grammar, for instance, do not provide LLMs with a mechanism to reduce verbosity while preserving the number of reasoning steps. In section 4.1.3, we show that these instructions lead to increased LLM uncertainty and higher semantic entropy as they attempt to reduce token count even if it means reducing number of reasoning steps, making each step more complex than CoT. While chain-of-drafts (Xu et al., 2025) allows serial computation, by limiting each step to 5 words they limit the vocabulary of the LLM, which means every step may not be informative. Chain-of-drafts does not perform well in small LLMs, which is acknowledged in the paper, and is corroborated by our findings in section 4.1.4.

3.3 TeleMathLang

As discussed, CoT increases a decoder-only transformer's ability to perform complex operations by increasing the depth of computation. While ar-

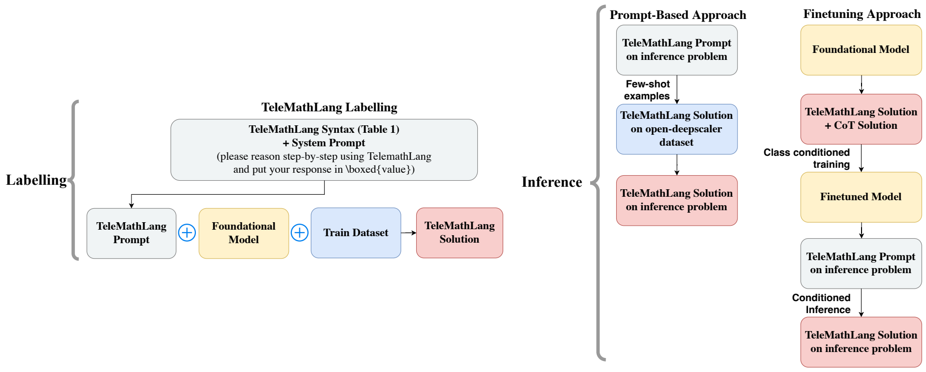

Figure 3: Overall Method Flow

chitecture depth is constant, CoT simulates depth increase by performing a shallow computation and putting the computation back into the LLM's context. Therefore, in order to maintain peformance gains from CoT, it is vital to **allow serial computation** for as many steps as required for the problem, with **sufficient vocabulary** for reasoning, i.e. the tokens chosen should be sufficient to ensure the shallow computation is exact. Our key insight is to provide a fixed set of tokens for forcing sequential computation while allowing LLMs to use any token required inside each thinking step.

To force sequential computation and thinking step, we develop on Kojy works such as Ye et al. (2025); Muennighoff et al. (2025); DeepSeek-AI et al. (2025), which show the association between the existence of self-correcting/self-reflecting tokens (wait, however) and logical connectors (therefore, since), and increased reasoning steps (and accuracy). We provide LLMs with TeleMathLang, which we define as a syntax with a **fixed set of reasoning keywords that can be used to begin a sentence**, and **rules to ensure reasoning keywords are used** (Table 1). These keywords are chosen to force thinking and sequential problem solving, avoiding the limitations of concise prompting instructions. We divide the keywords into core and extended, where core keywords are very commonly used and extended keywords are used in more complex problems.

Prompting In few-shot or zero shot setup with foundational models, the prompt provided consists of a description of TeleMathLang as an **ultra-minimal syntax**, the **keywords** and **rules**, few-shot **labelled** examples (or zero-shot, as is the case) and the instruction "Please think step-by-step in TeleMathLang, and return your final answer in boxed{answer}". The few-shot labelled examples are kept fixed across all evaluation benchmarks to assess generalizability of our method. The full prompt template and few-shot examples can be found in appendix B and C.

In section 4 we show that motivating TeleMathLang as an ultra-minimal syntax to LLMs consistently reduces token count while preserving accuracy. We also show that the idea of starting each sentence with a sequential reasoning keyword maintains low semantic entropy compared to other concise prompting techniques, implying that each thinking step is simpler. Figure 4 shows an example of CoT v/s TeleMathLang where we find that the number of thinking steps is higher in TeleMathLang while total token count is lower.

Finetuning We use TeleMathLang in a few-shot prompting setting as well as a finetuning setting. For finetuning, TeleMathLang responses are prepared for the train dataset using a state-of-the-art LLM, generally Claude 3.7 Sonnet (Anthropic,

Core Reasoning Keywords		Extended Reasoning Keywords		Syntax Rules
Given:	facts and setup	**Approach:**	strategy selection	Begin each line with one of
Let:	define variables	**Case:**	case analysis	the keywords
Then:	next logical step	**Insight:**	key observations	Use LaTeX for mathematical notation
So:	intermediate result	**Assume:**	for assumptions	(e.g., 2×x, (a + b), x²)
Check:	verify logic	**Lemma:**	supporting claims	Evaluate expressions when possible
Therefore:	conclusion	**Contradiction:**	proof method	(e.g., "Let x = 3×2 = 6"),
Verify:	double check	**Induction:**	inductive steps	showing calculation
Answer:	final result	**Generalize:**	extending patterns	

Table 1: TeleMathLang Syntax

Figure 4: CoT v/s TeleMathLang reasoning chains

2025). To leverage the familiarity of recent LLMs with CoT due finetuning on CoT samples, we carry out conditioned training (Kang et al., 2024). We create a train dataset comprising of CoT as well as TeleMathLang demonstrations, creating a class-conditioned fine-tuning dataset (with one class for CoT and one for TeleMathLang, distinguished by the associated prompt). The purpose of this is to allow LLMs to learn the correspondence between CoT and TeleMathLang reasoning. During inference, we provide the TeleMathLang prompt to get concise reasoning and response. More descriptions of labelling, training and testing are present in Section 4.

Figure 3 illustrates the overall flow with labelling, prompt-based algorithm and finetuning algorithm.

4 Experimental Setup and Results

4.1 Experimental Setup

Models and Benchmark Datasets We use three external benchmark datasets focusing on math and reasoning: MATH-500 (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021) and AI2-ARC (Clark et al., 2018). While GSM8k contains grade school math problems, MATH-500 is more challenging, containing competition math problems. ARC contains grade school science problems which require logic to solve. Combined, these benchmarks show an LLM's ability to reason, calculate and solve sequential problems.

Models To assess the generalizability of our approach across LLMs we run experiments across 0 shot, 4 shot and 8 shot setups on Claude 3.5 Sonnet v1 (Anthropic, 2024), Claude 3.7 Sonnet (Anthropic, 2025), Nova Pro (Intelligence, 2024) and Gemma 3 27B (Team et al., 2025) and report the average increase in accuracy and avg token reduction using TeleMathLang wrt CoT. The 4 shot and 8 shot examples were derived from the open-deepscaler dataset (Dang and Ngo, 2025) which contains challenging math problems not present in the 3 evaluation datasets.

Finetuning The Claude, Nova and Gemma models are instruction-tuned to be helpful and general purpose. In contrast, models such as the Qwen 2.5 instruct series are post-trained on large corpuses of data with responses in long-CoT format(Qwen et al., 2025). This is done to explicitly increase output length, and the models perform best when prompts match post-training data, and with increased test-time computation (Muennighoff et al., 2025). We show that prompting-based methods show poor performance here, and that this can be overcome with finetuning. For Qwen 2.5, we finetune the 1.5B instruction-tuned model in a conditional generation framework as discussed in section 3 and compare against CoT instructions as well as

Model	MATH-500			GSM8K			AI2-ARC			Average
	CoT	TML	Token↓	CoT	TML	Token↓	CoT	TML	Token↓	Token↓
Sonnet 3.5 v1	54.00%	**58.67%**	61.33%	89.66%	**95.48%**	69.21%	95.09%	94.65%	24.10%	51.55%
Sonnet 3.7	70.30%	**75.90%**	15.98%	95.98%	**96.29%**	24.76%	96.21%	95.88%	28.35%	23.03%
Nova Pro	72.87%	69.33%	46.52%	95.07%	93.91%	53.68%	93.87%	93.65%	46.33%	48.84%
Gemma 3 27B	76.50%	**78.50%**	35.88%	93.44%	93.33%	26.40%	92.30%	91.42%	32.46%	31.58%

Table 2: Performance comparison between CoT and TeleMathLang (TML) across datasets averaged across 0, 4 and 8 shot settings, showing accuracy and token reduction (Token↓).TeleMathLang maintains comparable accuracy while significantly reducing token count. Instances where it outperforms CoT are in bold.

Prompt Type	MATH-500		GSM8K	
	Acc.	Token↓	Acc.	Token↓
CoT	74.0%	0.00%	95.98%	0.00%
TeleMathLang	**75.9%**	15.98%	**96.29%**	24.76%
BeConcise	73.0%	34.51%	95.75%	30.52%
NoProperGrammar	72.2%	35.42%	96.05%	40.54%

Table 3: Comparison of prompting strategies on MATH-500 and GSM8K using Claude 3.7. Token↓ shows percentage reduction in token count compared to CoT.

CoT-based finetuning. The implementation details can be found in appendix D. We reviewed the licenses for all these datasets and models, and ensured that we stick to the intended usage of these for research purposes.

4.1.1 Performance Analysis

Overall performance results are shown in Table 2. The gains for some models in particular are noticeably large - Claude 3.5 v1 and Nova Pro achieve token count reductions of ∼50%+ over CoT while preserving accuracy. Accuracy is preserved in other models as well, with token count reductions of 20-35%. This shows that TeleMathLang consistently reduces the number of generated tokens across LLMs and benchmark datasets while maintaining comparable or improved accuracy.

4.1.2 TeleMathLang outperforms concise prompting instructions

Recent prompting strategies such as BeConcise, NoProperGrammar etc (Lee et al., 2025) have demonstrated the ability to reduce output token count without major loss in accuracy on simpler tasks. Table 3 compares them against TeleMathLang. All three minimal prompting approaches reduce token count. However, TeleMathLang uniquely preserves accuracy across both datasets. On GSM8K, a relatively less complex benchmark, all prompting methods maintain high performance, while on the more challenging MATH-500 benchmark, both BeConcise and NoProperGrammar show degraded performance as they prioritize brevity. TeleMathLang maintains high accuracy while token reduction is lesser. This aligns with the token complexity hypothesis (Lee et al., 2025) that problems possess an intrinsic **token complexity threshold**, below which reliable solution generation becomes statistically improbable. TeleMathLang allows LLMs to adaptively shorten output token count in response to the complexity of the task.

4.1.3 TeleMathLang allows models to perform concise reasoning with low semantic entropy

As discussed in Section 2, semantic entropy can be used to measure LLM response certainty and consistency. In this experiment, we compare the semantic entropy of TeleMathLang with 0 shot CoT, BeConcise and NoProperGrammar prompts.

Semantic Entropy Measurement We randomly sample the MATH-500 dataset and pick 100 problems. For each problem, we generate 20 responses from Claude 3.7 for each prompting approach. We then use gte-large (Li et al., 2023) to get sentence embeddings for the entire output. Finally, we calculate the pairwise distances of these embeddings and compare their means. Using sentence embeddings ensures that the focus remains on semantic meaning and not on individual tokens, and pairwise distances between similar lines of reasoning should be smaller than different lines of reasoning. The results are shown in Table 4.

We observe that TeleMathLang's mean pairwise distance is similar to CoT, while BeConcise and

Prompt Type	Mean Pairwise Distance	Increase Relative to CoT
CoT	0.048	0.00%
TeleMathLang	0.049	2.08%
BeConcise	0.057	18.75%
NoProperGrammar	0.065	35.42%

Table 4: Mean pairwise cosine distance between response embeddings across prompt types. CoT is used as the baseline.

Dataset	CoT		TeleMathLang		BeConcise		NoProperGrammar		Chain-of-Draft	
	Acc.	Token↓	Acc.	Token↓	Acc.	Token↓	Acc.	Token↓	Acc.	Token↓
AI2-ARC	55.85%	0.00%	44.30%	26.71%	28.42%	26.83%	17.05%	27.95%	6.68%	67.54%
GSM8K	72.23%	0.00%	67.12%	-4.71%	69.21%	24.24%	69.14%	28.94%	51.18%	40.08%

Table 5: Performance comparison of concise promp-based strategies using Qwen 2.5 1.5B. Each cell shows accuracy and token reduction relative to CoT.

Dataset	CoT (non-FT)		CoT (FT)		TeleMathLang (FT)		Conditional Gen.	
	Acc.	Token↓	Acc.	Token↓	Acc.	Token↓	Acc.	Token↓
AI2-ARC	55.85%	0.00%	72.24%	8.18%	70.23%	40.06%	**74.92%**	38.98%
GSM8K	**71.86%**	0.00%	71.86%	2.07%	69.85%	52.82%	71.36%	46.29%

Table 6: Qwen-2.5-1.5B on AI2-ARC and GSM8K with various inference strategies. Token↓ shows percentage reduction in generated tokens relative to non-fine-tuned CoT.

NoProperGrammar result in significantly higher distances. Providing a syntax of reasoning keywords ensure LLMs perform as many reasoning steps as required, allowing them to be more consistent in their outputs, while general concise prompting instructions may lead to increased uncertainty as LLMs try to perform complex operations in a single step.

4.1.4 TeleMathLang reduces token count even in models post-trained to generate long output

As mentioned in section 2, recent advancements especially in math benchmarks are largely attributable to significant investments on post-training on CoT datasets. Due to the usage of CoT in SFT, models such as Qwen 2.5 are optimized only in setups where model is asked to generate long step-by-step reasoning. Restricting these models through instructions hampers performance, and we noticed that prompt-based strategies (BeConcise, NoProperGrammar, chain-of-draft) do not work in these models(Table 5).

We finetuned Qwen 2.5 1.5B using pure TeleMathLang samples as well as combined samples for conditional generation, and compared it with CoT prompting as well as finetuning. We find that this helps the model learn to perform concise reasoning without compromizing on quality, overcoming the inflexibility of the base model. Conditional generation shows the best performance, as the model learns to associate TeleMathLang solutions with CoT since both solutions are present in the training data. The results are shown in table 6.

5 Conclusion

In this work, we proposed TeleMathLang, a minimal syntax with reasoning keywords and rules that enables LLMs to generate complete chains-of-thought while reducing total token count. We showed that it generalizes well across instruction-tuned LLMs, and requires light finetuning in models which have been explicitly trained to generate longer chains of thought. It showed significant reduction in output token count without compromising accuracy, adaptively changing CoT length according to the token complexity of problems. Finally, the specified keywords and syntactical framework allows LLMs to generate output with low semantic entropy (representing uncertainty) compared to other minimal prompting techniques. The combination of low resource requirement and no loss in accuracy positions TeleMathLang as a promising direction of research, and future work could focus on optimizing reasoning tokens as well as reducing token count within each reasoning step.

Limitations

We acknowledge the limitations of TeleMathLang with a view to motivating further research in this field. TeleMathLang-syntax based prompts generalize well across LLMs, except for models heavily post-trained on problems with CoT solutions. Fine-tuning such models may require labelled samples, which can be generated using other LLMs, but care needs to be taken with respect to their accuracy. Additionally, we provide a set of keywords beyond just "wait" as in (Muennighoff et al., 2025), but more investigation needs to be done in future works to understand the most optimal reasoning keywords using more diverse reasoning datasets, to avoid the risk of disallowing certain critical logical operations. Finally, we will explore reducing token count further within reasoning steps through more explicit instructions.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *Preprint*, arXiv:2503.04697.

Anthropic. 2024. claude-3-5-sonnet.

Anthropic. 2025. Claude 3.7 sonnet and claude code.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Preprint*, arXiv:2211.12588.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

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. *arXiv preprint arXiv:2110.14168*.

Quy-Anh Dang and Chris Ngo. 2025. Reinforcement learning for reasoning in small llms: What works and what doesn't. *Preprint*, arXiv:2503.16219.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630.

Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and

Kartik Talamadupula. 2025. Concise reasoning via reinforcement learning. *Preprint*, arXiv:2504.05185.

Leo Gao, John Schulman, and Jacob Hilton. 2022. Scaling laws for reward model overoptimization. *Preprint*, arXiv:2210.10760.

Yiding Hao, Dana Angluin, and Robert Frank. 2022. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10:800–810.

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. *Preprint*, arXiv:2103.03874.

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. 2025. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. *Preprint*, arXiv:2504.07086.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *Preprint*, arXiv:2504.01296.

HuggingFaceH4. 2024. aime_2024.

Amazon Artificial General Intelligence. 2024. The amazon nova family of models: Technical report and model card. *Amazon Technical Reports*.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. *Preprint*, arXiv:2401.04925.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. 2024. C3ot: Generating shorter chain-of-thought without compromising effectiveness. *Preprint*, arXiv:2412.11664.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. 2024. Semantic entropy probes: Robust and cheap hallucination detection in llms. *Preprint*, arXiv:2406.15927.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. 2023. Measuring faithfulness in chain-of-thought reasoning. *Preprint*, arXiv:2307.13702.

Ayeong Lee, Ethan Che, and Tianyi Peng. 2025. How well do llms compress their own chain-of-thought? a token complexity approach. *Preprint*, arXiv:2503.01141.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *Preprint*, arXiv:2308.03281.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024. Chain of thought empowers transformers to solve inherently serial problems. *Preprint*, arXiv:2402.12875.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *Preprint*, arXiv:2503.20783.

Aman Madaan, Katherine Hermann, and Amir Yazdanbakhsh. 2023. What makes chain-of-thought prompting effective? a counterfactual study. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1448–1535, Singapore. Association for Computational Linguistics.

William Merrill and Ashish Sabharwal. 2023. The parallelism tradeoff: Limitations of log-precision transformers. *Preprint*, arXiv:2207.00729.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Qwen. 2024. Qwen generation config.

C. E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil

Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint*.

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.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *Preprint*, arXiv:2502.18600.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *Preprint*, arXiv:2502.03387.

A Boolean Circuit Complexity Classes

A **family** of circuits C_n is of constant depth if there exists a constant K such that the depth of C_n is bounded by K for all n. A family of circuits is of polynomial size if there exists a constant c such that the size of C_n is bounded by nc + c for all n. The set AC^0 is the set of families of Boolean circuits of both constant depth and polynomial size (Hao et al., 2022). The set TC^0 is a slightly larger set of circuit families that include threshold gates. These sets, with varying depths, sizes and operations, are used to represent complexity of computations - for instance, an operation like majority voting is too complex for AC^0 but belongs to TC^0.

B TeleMathLang prompt template

TeleMathLang Prompt
<SYSTEM> Please reason step by step using TeleMathLang, and put your final answer within \boxed{value} <USER> You solve math problems using ultra-minimal TeleMathLang language. Here's the syntax syntax: <syntax in Table 1> Here's a few examples: <few-shot> Problem: <problem>

Table 7: TeleMathLang Prompt

C Few-shot examples

Here we share a few few-shot TeleMathLang examples. We provide the CoT solution side-by-side **for reference only**. The actual few shot examples included in the prompt **only include the TeleMathLang solutions**.

Example 1: Square Window Problem

Problem: Doug constructs a square window using 8 equal-size panes of glass. The ratio of the height to width for each pane is 5 : 2, and the borders around and between the panes are 2 inches wide. In inches, what is the side length of the square window?

Chain-of-Thought (CoT) Reasoning

1. Identify the dimensions of each pane: Given that the ratio of the height to the width of each pane is $5 : 2$, let the height of each pane be $5x$ inches and the width be $2x$ inches.

2. Calculate the total dimensions of the window: The window is constructed with 8 panes arranged in 2 rows and 4 columns. The borders between and around the panes are 2 inches wide.

3. Calculate the total width of the window:

- There are 4 panes in each row, each with a width of $2x$ inches.
- There are 3 borders of 2 inches between panes and 2 borders on the sides.
- Total width = $4(2x) + 5(2) = 8x + 10$

4. Calculate the total height of the window:

- There are 2 panes in each column, each with a height of $5x$ inches.
- There is 1 border between panes and 2 borders at the top and bottom.
- Total height = $2(5x) + 3(2) = 10x + 6$

5. Set up the equation for the square: Since the window is square,

$$8x + 10 = 10x + 6$$

6. Solve for x:

$$\begin{aligned} 8x + 10 &= 10x + 6 \\ 10 &= 2x + 6 \\ 4 &= 2x \\ x &= 2 \end{aligned}$$

7. Calculate side length:

$$8(2) + 10 = 16 + 10 = 26 \text{ inches}$$

8. Final Answer:

$$\boxed{26}$$

TeleMathLang Reasoning

- Given: Square window, 8 panes, pane ratio 5:2 (h:w), 2-inch borders
- Let: Pane height = $5x$, width = $2x$
- Then: Window layout = 2 rows $\times$ 4 columns
- Then: Total width = $4(2x) + 5(2) = 8x + 10$
- Then: Total height = $2(5x) + 3(2) = 10x + 6$
- Check: $8x + 10 = 10x + 6$
- Then: $x = 2$
- Then: Side length = 26
- Verify: $26 = 26$
- **Answer:** $\boxed{26}$

Example 2: Polynomial Interpolation Problem

Problem: Let $P(x)$ be a polynomial of degree $3n$ such that:

$$\begin{aligned} P(3k) &= 2, \\ P(3k-2) &= 1, \\ P(3k-1) &= 0, \quad \text{for } k = 1, \dots, n \end{aligned}$$

and $P(3n+1) = 730$. Find n.

Chain-of-Thought (CoT) Reasoning

To solve for n, we analyze the polynomial $P(x)$ using Lagrange Interpolation.

1. **Define points:**
   - $P(3k) = 2$ for $k = 0, 1, \dots, n$
   - $P(3k-2) = 1$, $k = 1, \dots, n$
   - $P(3k-1) = 0$, $k = 1, \dots, n$

2. **Interpolation:**

$$P(x) = 2\sum_{p=0}^{n} \prod_{\substack{0 \le r \le 3n \\ r \ne 3p}} \frac{x-r}{3p-r} + \sum_{p=1}^{n} \prod_{\substack{0 \le r \le 3n \\ r \ne 3p-2}} \frac{x-r}{3p-2-r}$$

3. **Evaluate at $x = 3n+1$:**

$$P(3n+1) = 2\sum_{p=0}^{n} \prod_{\substack{0 \le r \le 3n \\ r \ne 3p}} \frac{3n+1-r}{3p-r} + \sum_{p=1}^{n} \prod_{\substack{0 \le r \le 3n \\ r \ne 3p-2}} \frac{3n+1-r}{3p-2-r}$$

4. **Simplify with alternating binomial sum:**

$$P(3n+1) = \left(\binom{3n+1}{0} - \binom{3n+1}{3} + \binom{3n+1}{6} - \cdots\right)(2(-1)^{3n}-1)+1$$

5. **Given: $P(3n+1) = 730$:**

$$(\text{alternating sum})\,(2(-1)^{3n}-1) = 729$$

6. **Use lemma:**

$$\frac{2^{3n+2} \cdot \sin^{3n+1}\left(\frac{\pi}{3}\right)}{3} = 729 \Rightarrow n = 4$$

7. **Conclusion:**

$$\boxed{n = 4}$$

TeleMathLang Reasoning

Given: $P(x)$ of degree $3n$
Let: $P(3k) = 2, P(3k-2) = 1, P(3k-1) = 0$ for respective ranges
Let: $P(3n+1) = 730$
Approach: Lagrange interpolation
Then:

$$P(x) = 2\sum_{p=0}^{n} \prod_{r \ne 3p} \frac{x-r}{3p-r} + \sum_{p=1}^{n} \prod_{r \ne 3p-2} \frac{x-r}{3p-2-r}$$

So:

$$P(3n+1) = \left(\binom{3n+1}{0} - \binom{3n+1}{3} + \cdots\right) \cdot (2(-1)^{3n}-1)+1$$

Check: Matches given $P(3n+1) = 730$
Solve: Alternating binomial identity
Lemma: $\frac{2^{3n+2}\sin^{3n+1}(\pi/3)}{3} = 729 \Rightarrow n = 4$
Answer: $\boxed{4}$

D Finetuning Implementation Details

Implementation details We use Qwen's recommended settings (temperature, repetition penalty etc) shared by Qwen developers (Qwen, 2024) with vllm for inference, and for finetuning, we use huggingface's TRL library. We evaluate the performance of Conditional Generation against CoT w/o fine tuning, CoT with fine-tuning, and solely TeleMathLang fine-tuning. The model is trained for 10 epochs on a train dataset sample of size 1000, with learning rate $5e^{-5}$ for TeleMathLang and CoT, and for 5 epochs with combined 2000 samples for conditional generation