ProofSketch: Efficient Verified Reasoning for Large Language Models

Disha Sheshanarayana*
Manipal University Jaipur
disha.229301161@muj.manipal.edu

Tanishka Magar*
Manipal University Jaipur
tanishka.229301736@muj.manipal.edu

Abstract

Reasoning methods such as chain-of-thought prompting and self-consistency have shown immense potential to improve the accuracy of large language models across various reasoning tasks. However such methods involve generation of lengthy reasoning chains, which substantially increases token consumption, computational cost, and latency. To address this inefficiency, we propose ProofSketch, a verification-guided reasoning framework that integrates symbolic closure computation, lexicographic verification and adaptive sketch generation. Our experiments show that ProofSketch consistently reduces token usage while improving accuracy, demonstrating that this approach offers a promising path for efficient and trustworthy reasoning. The code is available at https://github.com/tanishka66/ProofSketch.

1 Introduction

The reasoning capabilities of LLMs have been explored, analysed and experimented with. Research on LLM reasoning spans prompting-based strategies, structured search methods, and efficiency-oriented decoding, each offering different trade-offs between accuracy, interpretability, and cost. However, these approaches typically require the model to generate longer reasoning chains. While this can improve accuracy, it also comes at the cost of substantial token usage and higher latency, reducing their efficiency in settings where computational resources are limited.

It has been seen that in many cases, LLMs generate unnecessarily long and elaborate reasoning chains, even for trivial questions, often referred to as the overthinking problem [17]. This leads to significant wastage of computational resources. Previous works have attempted to improve upon this by enforcing constraints, using instruction tuning and dynamically adjusting reasoning length based on problem difficulty to control output lengths [6, 19]. Another limitation is that intermediate reasoning steps remain unchecked. [10] Long reasoning traces provide no guarantee of logical validity. As a result, we cannot be certain whether the final prediction is grounded in valid reasoning or reflect the errors propagated through fluent but incorrect intermediate steps. Combined, these challenges often cause excessive reasoning length and lack of intermediate verification. This highlights the need for methods that can achieve accurate reasoning under tight compute budgets while also providing correctness guarantee.

To address this gap we propose ProofSketch, a verification-guided efficient reasoning framework. Instead of generating lengthy reasoning chains, our method produces multiple short "sketches" containing atomic claims, then selects the sketch with maximum verification coverage, thus enabling low-cost yet reliable shallow reasoning.

*These authors contributed equally to this work

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Efficient Reasoning.

2 Related Works

Building on the growing interest in improving LLM reasoning, scientists have proposed techniques such as chain-of-thought (CoT) prompting, self-consistency, and proof generation, which have achieved notable increases in accuracy [14, 13, 16], but computational costs remain excessive. To overcome this studies have explored reasoning under explicit budget constraints applying token-aware optimization strategies [6], adaptive control of reasoning length via confidence probing [5], and dynamic token budgeting that adjusts the reasoning process per instance [8]. Other methods to reduce output lengths include replacing token-heavy CoT traces with soft latent vectors [15] or leveraging reward shaping to encourage more concise reasoning.

The study [18] showed how reducing tasks to minimal (atomic) claims can improve soundness and interpretability, which directly inspired `ProofSketch`. Works [1] and [12] have explored sketching-based methods where shorter reasoning structures are generated, successfully reducing token usage. Studies have also explored ways to ensure correctness in responses. One of them which stands out is verification-guided reasoning, where intermediate steps are explicitly checked [16, 9, 2, 3]. ProofSketch combines these strands to ensure verified, token-efficient reasoning.

3 `ProofSketch`

3.1 Problem Formulation

Let $\mathcal{T}$ be a logical theory containing facts and rules, and Q be a question requiring True/False/Unknown classification. Given the computational constraints of modern language models, we seek reasoning methods that simultaneously optimize multiple objectives: generating accurate answers, minimizing computational overhead, and providing formal guarantees about reasoning correctness.

We define the efficiency-accuracy-certification optimization problem as: $\max \text{Accuracy}(f(\mathcal{T}, Q))$ s.t. $\mathbb{E}[\text{tokens}(f(\mathcal{T}, Q))] \leq \beta$, $\text{Cert}(f(\mathcal{T}, Q)) \to 1$ where f is our reasoning function, β is the token budget constraint, and Cert measures the proportion of formally verified reasoning steps. This multi-objective formulation captures the fundamental challenge in neural reasoning: balancing accuracy, efficiency, and reliability while providing formal guarantees about reasoning processes.

3.2 `ProofSketch` Framework

`ProofSketch` introduces a novel reasoning framework that integrates symbolic closure computation with verifier-gated neural generation. The framework operates through a multi-stage pipeline designed to optimize accuracy, efficiency, and certification simultaneously, as illustrated in Figure 1.

Symbolic Closure Foundation The framework begins by parsing theory $\mathcal{T}$ into positive facts F_+, negative facts F_-, and logical rules R. Forward chaining derives the symbolic closure $C(\mathcal{T}) = \text{FC}(F_+, F_-, R)$, which serves as the foundation for both direct answer checking and claim verification. This closure enables immediate resolution of questions derivable through pure logic while providing a verification oracle for generated content.

Verifier-Gated Generation When generation is required, the framework samples up to $K{=}4$ short *sketch* candidates under an adaptive token budget (120 tokens if $C(\mathcal{T})$ already contains facts about the queried entity, 160 otherwise) with mild temperature ($\tau{=}0.3$) for controlled diversity. Each sketch is emitted in a structured, machine-readable format consisting of a proposed answer to the query Q together with a small set of atomic declarative claims; each claim must conform to the canonical unary form "e is a" or "e is not a", where e is an entity and a is an attribute. We canonicalize surface mentions so that entities and attributes align with the symbols in $\mathcal{T}$ and reduce the sketch to a minimal anchored subset that refers directly to the entity named in Q, yielding a compact, query-focused justification rather than an arbitrary chain-of-thought. Because model outputs may be imperfectly structured, we apply a lightweight repair pass prior to parsing; if a sampled sketch cannot be reliably parsed into a candidate answer plus at least one canonical claim, it is treated as having no usable claims and is de-prioritized by the verifier-guided selector (i.e., it cannot be marked certified downstream).

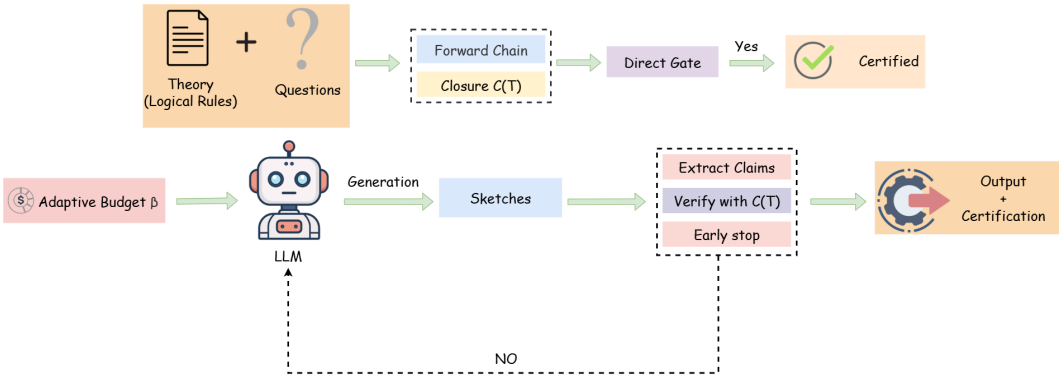

Figure 1: `ProofSketch` framework generates sketches from theories and questions, applies closure verification through forward chaining, and uses adaptive budgeting for re-generation when verification fails. Successful verification produces certified outputs, while failures trigger budget-controlled iterative refinement until certification is achieved.

Multi-Objective Verification and Selection Generated sketches are evaluated through formal verification against $C(\mathcal{T})$. The framework employs lexicographic scoring that prioritizes: (1) full certification (all claims verified), (2) partial verification coverage, (3) token efficiency, and (4) consistency with closure decisions. Early stopping occurs when a fully certified sketch is found, providing computational savings while ensuring formal correctness guarantees.

Certification and Output The framework produces three key outputs: a final answer (with closure correction if needed), a set of formally verified atomic claims supporting the reasoning, and a certification status indicating the degree of formal verification achieved. This design enables transparent reasoning assessment and post-hoc analysis of model decisions.

4 Experimentation

4.1 Experimental Setup

Datasets. We evaluate `ProofSketch` on a curated subset of the ProofWriter dataset [11], a widely-used logical reasoning benchmark containing theories expressed in natural language with corresponding True/False/Unknown questions. We collected 300 data points from ProofWriter, selected to represent varying reasoning depths and complexity levels to ensure comprehensive evaluation of our symbolic-neural integration approach.

Baselines. We evaluate three prompting approaches across three language models: Mistral-7B [7], DeepSeek-R1 Distill [4] Llama-8B, and Qwen-7B. The prompting approaches include: (1) **Zero-shot** prompting for immediate True/False/Unknown classification without reasoning steps, (2) **Short CoT** with up to 3 concise reasoning lines before the final answer, and (3) **Long CoT** allowing up to 10 detailed reasoning steps.

Metrics. We measure accuracy on True/False/Unknown classifications, certification rate (percentage of formally verified responses), mean token usage, P95 token consumption, and average latency.

Implementation Details. `ProofSketch` uses $K = 4$ sketch candidates with adaptive token budgets ($\beta_1 = 120$ for entities in symbolic closure, $\beta_2 = 160$ otherwise) and temperature $\tau = 0.3$ for controlled diversity. The forward-chaining engine computes symbolic closures using first-order logic predicates. Multi-objective scoring employs lexicographic ordering across certification status, verification coverage, token efficiency, and response consistency.

4.2 Results

We evaluate `ProofSketch` across three language models on a 300 example ProofWriter dataset, measuring accuracy, token efficiency, and formal verification capabilities. Table 1 presents our

comprehensive evaluation results, demonstrating that `ProofSketch` achieves strong reasoning performance across all models: 68.0% accuracy with R1-Distill-Llama-8B, 52.0% with Mistral-7B, and 54.0% with R1-Distill-Qwen-7B. Beyond competitive accuracy, a key distinguishing feature of our method is its formal verification capability, entirely absent in standard prompting approaches. The method achieves remarkable certification rates of 42.0% with R1-Llama and R1-Qwen, and 84.0% with Mistral-7B, representing responses that receive complete mathematical verification through our symbolic closure system. Regarding computational efficiency, `ProofSketch` demonstrates exceptional performance with Mistral-7B, requiring only 27.96 tokens per query on average, R1-Qwen achieving similar efficiency at 30.28 tokens, while the R1-Llama configuration uses 137.94 tokens but delivers the highest accuracy. These results validate our core hypothesis that symbolic preprocessing and verifier-gated generation enable both competitive reasoning performance and formal verification capabilities, establishing a new paradigm for trustworthy neural reasoning systems while maintaining computational efficiency across different model architectures. We have analyzed the detailed findings in Appendix B. To assess the generalizability of our findings, we additionally conduct evaluations on a larger 1,000 example ProofWriter dataset, with full results reported in Appendix C.

Table 1: Comparison of reasoning methods on three models (Deepseek-R1-Distill-Llama-8B, Mistral-7B, Deepseek-R1-Distill-Qwen-7B), reporting Accuracy(Acc), Mean tokens(Tok), and Certified fraction(Cert).

	R1-Distill-Llama-8B			Mistral-7B			R1-Distill-Qwen-7B		
Method	Acc	Tok	Cert	Acc	Tok	Cert	Acc	Tok	Cert
Zero-shot	0.37	**122.27**	0	0.33	**7.00**	0	0.39	**9.85**	0
Short-CoT	0.52	214.83	0	0.48	52.86	0	0.44	48.75	0
Long-CoT	0.52	218.71	0	0.41	101.76	0	0.47	101.09	0
`ProofSketch`	**0.68**	137.94	**0.42**	0.52	27.96	**0.84**	**0.54**	30.28	**0.42**

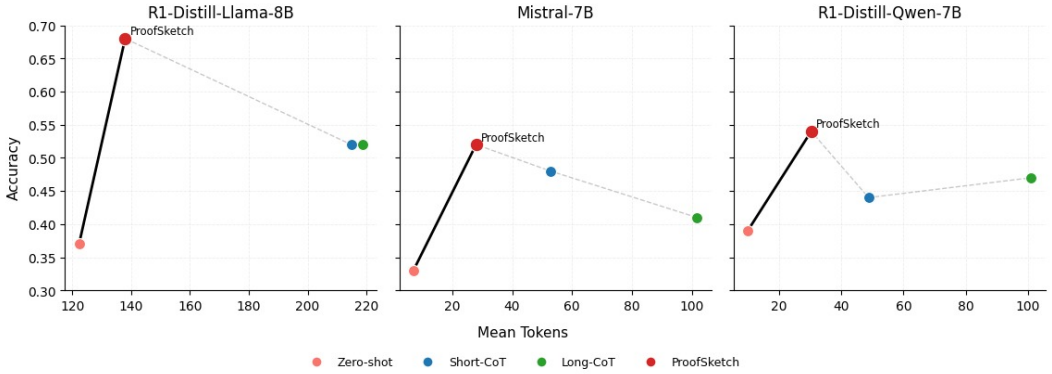

Figure 2: Pareto Frontier for Accuracy vs Token Usage across Models

4.2.1 Comparison of Latency

Latency results across all methods and models are provided in Table 2. While ProofSketch consistently reduced token usage compared to other prompting techniques, our results reveal that this efficiency sometimes could lead to a noticeable latency overhead. Our framework remained competitive for most and even significantly reduced on Mistral-7B. The only notable increase in the latency overhead was for R1-Distill-Llama-8B. We did not identify a single dominant cause for this increase, but we hypothesize that the cumulative costs of multiple sketch generations, closure computation, and verification checks outweigh the generation savings in some cases. Future work could explore caching closure computations and parallelizing sketch generation to mitigate and control this overhead.

Table 2: Mean latency (ms) across reasoning strategies and models.

Method	Distill-Llama-8B	Mistral-7B	Distill-Qwen-7B
Zero-shot	9240.05	1018.62	1340.26
Short-CoT	15908.20	7581.99	6926.19
Long-CoT	15984.95	11069.88	9909.83
`ProofSketch`	31741.47	5593.77	11153.46

5 Conclusion

(1) Conclusion. In this work we propose `ProofSketch`, a novel framework designed to ensure efficient and verified reasoning of large language models by combining symbolic closure, adaptive sketch generation, and lexicographic verification. ProofSketch involves the generation of multiple short sketches with atomic claims, which are then verified to select the most reliable sketch. This directly addresses two key limitations of prior approaches: longer reasoning lengths and unverified intermediate steps and thus offers a promising direction for deploying LLMs in compute-constrained or tightly budgeted settings. **(2) Limitations.** The additional verification stage in `ProofSketch` introduces a modest latency overhead compared to purely generative baselines. A key limitation of `ProofSketch` is that it relies on simple symbolic checks, which may not scale to more complex reasoning domains. Furthermore, it has only been tested on controlled datasets, meaning its effectiveness in real-world noisy environments remains to be determined. **(3) Future Work.** Future work could involve extending this framework to more complex domains, exploring adaptive sketch generation policies, and integrating neural verifiers for broader coverage.

References

[1] Simon A. Aytes, Jinheon Baek, and Sung Ju Hwang. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *arXiv preprint arXiv:2503.05179*, 2025.

[2] Lang Cao. Graphreason: Enhancing reasoning capabilities of large language models through a graph-based verification approach. In *Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ACL 2024)*, pages 1–12, Bangkok, Thailand, 2024. Association for Computational Linguistics.

[3] Jishnu Ray Chowdhury and Cornelia Caragea. Zero-shot verification-guided chain of thoughts. *arXiv preprint arXiv:2501.13122*, 2025.

[4] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[5] Yichao Fu, Junda Chen, Yonghao Zhuang, Zheyu Fu, Ion Stoica, and Hao Zhang. Reasoning without self-doubt: More efficient chain-of-thought through certainty probing. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025.

[6] Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24860, 2025.

[7] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[8] Zheng Li, Qingxiu Dong, Jingyuan Ma, Di Zhang, Kai Jia, and Zhifang Sui. Selfbudgeter: Adaptive token allocation for efficient llm reasoning. *arXiv preprint arXiv:2505.11274*, 2025.

[9] Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. Deductive verification of chain-of-thought reasoning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[10] Minh-Vuong Nguyen, Linhao Luo, Fatemeh Shiri, Dinh Phung, Yuan-Fang Li, Thuy-Trang Vu, and Gholamreza Haffari. Direct evaluation of chain-of-thought in multi-hop reasoning with knowledge graphs. *arXiv preprint arXiv:2402.11199*, 2024.

[11] Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *arXiv preprint arXiv:2012.13048*, 2020.

[12] Jikai Wang, Juntao Li, Jianye Hou, Bowen Yan, Lijun Wu, and Min Zhang. Efficient reasoning for llms through speculative chain-of-thought. *arXiv preprint arXiv:2504.19095*, 2025.

[13] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Nazneen Sharan, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[14] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[15] Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23336–23353. Association for Computational Linguistics, 2025.

[16] Kaiyu Yang, Jia Deng, and Danqi Chen. Generating natural language proofs with verifier-guided search. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 89–105, 2022.

[17] Linan Yue, Yichao Du, Yizhi Wang, Weibo Gao, Fangzhou Yao, Li Wang, Ye Liu, Ziyu Xu, Qi Liu, Shimin Di, and Min-Ling Zhang. Don't overthink it: A survey of efficient r1-style large reasoning models. *arXiv preprint arXiv:2508.02120*, 2025.

[18] Yuji Zhang, Qingyun Wang, Cheng Qian, Jiateng Liu, Chenkai Sun, Denghui Zhang, Tarek Abdelzaher, Chengxiang Zhai, Preslav Nakov, and Heng Ji. Atomic reasoning for scientific table claim verification. *arXiv preprint arXiv:2506.06972*, 2025.

[19] Kai Zhao, Yanjun Zhao, Jiaming Song, Shien He, Lusheng Zhang, Qiang Zhang, and Tianjiao Li. Saber: Switchable and balanced training for efficient llm reasoning. *arXiv preprint arXiv:2508.10026*, 2025.

A `ProofSketch` Algorithm

Algorithm 1 `ProofSketch` ProofSketch with Verifier-Gated Decoding

Input: Theory $\mathcal{T}$, question Q, sketches $K = 4$, budgets $\beta_1 = 120, \beta_2 = 160$
Output: Answer $\hat{a}$, verified claims $\hat{C}$, certification status σ

Parse $\mathcal{T}$ into facts (F_+, F_-) and rules R
$C(\mathcal{T}) \leftarrow \text{ForwardChain}(F_+, F_-, R)$
if $Q \in C(\mathcal{T})$ **then**
    return $(\text{closure_answer}(Q), \emptyset, \text{CERTIFIED})$
end if
$\beta \leftarrow \beta_1$ if $\text{entity}(Q) \in C(\mathcal{T})$ else β_2
for $k = 1, 2, \ldots, K$ **do**
    $s_k \leftarrow \text{LLM}(\text{prompt}(\mathcal{T}, Q), \beta, \tau)$
    $(\text{claims}_k, \text{answer}_k) \leftarrow \text{parse_json}(s_k)$
    $\text{verdicts}_k \leftarrow [\text{verify}(c, C(\mathcal{T})) \mid c \in \text{claims}_k]$
    $\text{score}_k \leftarrow (\text{cert}_k, |\text{verified}_k|, -\text{tokens}(s_k), \text{consistency}_k)$
    if $\text{cert}_k = 1$ **then**
        return $(\text{answer}_k, \text{claims}_k, \text{CERTIFIED})$
    end if
end for
$k^* \leftarrow \arg\max_{\text{lex}}(\text{score}_k)$
if $C(\mathcal{T}) \models Q$ **then**
    $\hat{a} \leftarrow \text{closure_answer}(Q)$
else
    $\hat{a} \leftarrow \text{answer}_{k^*}$
end if
return $(\hat{a}, \text{verified_claims}_{k^*}, \text{certification_status})$

B Statistical Analysis

To demonstrate `ProofSketch`'s efficiency advantages, we analyzed token savings compared to Long-CoT baselines across all three models. Figure 3 shows that ProofSketch achieves substantial computational savings: 37.0% token reduction on R1-Distill-Llama-8B, 69.6% on Mistral-7B, and an impressive 71.0% on R1-Distill-Qwen-7B. Notably, the Qwen-7B model demonstrates the highest efficiency gains, indicating optimal compatibility with our symbolic verification approach.

To validate our adaptive budgeting mechanism, we conducted an ablation study on R1-Distill-Qwen-7B examining the impact of fixed sketch budgets versus our adaptive allocation strategy. Figure 4 reveals that fixed budget constraints lead to suboptimal performance across the entire budget range (120-220 tokens), with accuracy consistently remaining below the adaptive approach. This demonstrates that our dynamic budget allocation is crucial for optimal performance, as it intelligently adjusts computational resources based on problem complexity rather than applying rigid constraints. These findings confirm that `ProofSketch`'s integrated approach delivers superior efficiency while maintaining reasoning quality through intelligent resource management.

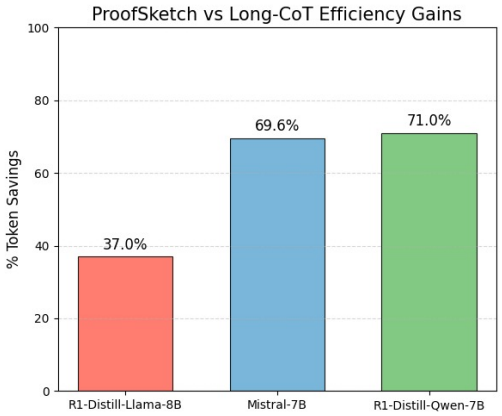

Figure 3: Token Savings

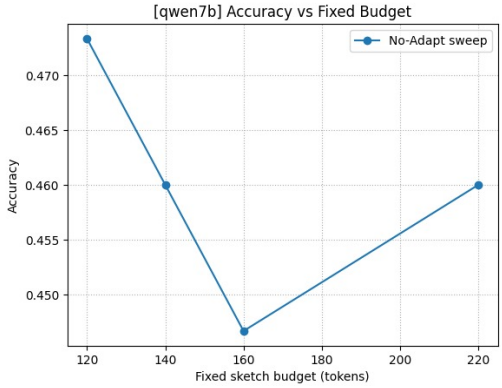

Figure 4: Ablation Study

C Extended Evaluation

To assess the generalizability of our findings, we further evaluate `ProofSketch` on an extended 1,000 example subset ProofWriter dataset. We saw that the overall performance trends remained consistent, thus confirming that the `ProofSketch` framework can scale on a larger dataset effectively without degradation in accuracy and tokens. These results suggest that the observed benefits of `ProofSketch` are not limited to small-scale benchmarks but extend to broader settings as well.

Table 3: Evaluation of `ProofSketch` across larger dataset

Method	Accuracy	Mean Tokens	Mean Latency
Zero-shot	0.394	9.598	**1312.06**
Short-CoT	0.404	49.735	7042.52
Long-CoT	0.424	98.126	10012.91
`ProofSketch`	**0.496**	**28.622**	9468.45