ECONPROVER: Towards More Economical Test-Time Scaling for Automated Theorem Proving

Mukai Li[†,1,2], **Linfeng Song**[†,1], **Zhenwen Liang**[1], **Jiahao Xu**[1], **Shansan Gong**[2], **Qi Liu**[†,2], **Haitao Mi**[1], **Dong Yu**[1]

[1]Tencent [2]The University of Hong Kong

kaikiaia3@gmail.com, lfsong@global.tencent.com, liuqi@cs.hku.hk

Abstract

Large Language Models (LLMs) have recently advanced the field of Automated Theorem Proving (ATP), attaining substantial performance gains through widely adopted test-time scaling strategies, notably reflective Chain-of-Thought (CoT) reasoning and increased sampling passes. However, they both introduce significant computational overhead for inference. Moreover, existing cost analyses typically regulate only the number of sampling passes, while neglecting the substantial disparities in sampling costs introduced by different scaling strategies. In this paper, we systematically compare the efficiency of different test-time scaling strategies for ATP models and demonstrate the inefficiency of the current state-of-the-art (SOTA) open-source approaches. We then investigate approaches to significantly reduce token usage and sample passes while maintaining the original performance. Specifically, we propose two complementary methods that can be integrated into a unified **EconRL** pipeline for amplified benefits: (1) a dynamic Chain-of-Thought (CoT) switching mechanism designed to mitigate unnecessary token consumption, and (2) Diverse parallel-scaled reinforcement learning (RL) with trainable prefixes to enhance pass rates under constrained sampling passes. Experiments on miniF2F and ProofNet demonstrate that our **ECONPROVER-GD** achieves comparable performance to baseline methods with only 12% of the computational cost. This work provides actionable insights for deploying lightweight ATP models without sacrificing performance.

1 Introduction

Automated theorem proving (ATP) aims to automatically generate formal proofs for given mathematical statements. By transforming natural language statements into theorems in a formal language, e.g., Lean (Moura and Ullrich, 2021) or Isabelle (Wenzel et al., 2008), and interacting with the corresponding engine to construct full proofs, an ATP system generates machine-verified proofs that guarantee strict logical correctness. The integration of large language models (LLMs) into ATP has marked a paradigm shift in formal mathematics, with systems like DeepSeek-Prover (Ren et al., 2025; Lin et al., 2025b), Kimina-Prover (Wang et al., 2025), and Goedel-Prover (Lin et al., 2025a,b) achieving unprecedented success rates on challenging benchmarks such as miniF2F (Zheng et al., 2022) and PutmanBench (Tsoukalas et al., 2024). In addition to the improved reasoning capability of the base LLMs, recent advances in ATP predominantly rely on test-time scaling strategies, which can be categorized into two approaches: sequential scaling and parallel scaling. Inspired by the success of reflective long CoT (DeepSeek-AI, 2025) in general reasoning tasks, **sequential scaling** augments the proving process with reflective CoT reasoning that mimic how human solves problems. Typical examples are DeepSeek-Prover-V2 (Ren et al., 2025), which uses natural language sketches to decompose the main theorem into smaller subgoals, and Kimina-Prover (Wang et al., 2025), which writes interleaved reasoning and proving blocks to underscore the proving strategy in the macro level. On the other hand, **parallel scaling** improves performance by increasing the number of independent sampling passes. Following its widespread success in verifiable tasks like code generation and mathematical reasoning, parallel scaling has been widely adopted by ATP models to boost proving performance.

Despite their apparent effectiveness, current state-of-the-art ATP models exhibit significant inefficiencies in their test-time scaling methods. Sequential scaling techniques typically result in a 10-15 times increase in token usage compared to approaches that do not incorporate reflective CoT reasoning. Some configurations (Ren et al., 2025) require as much as 8192 proving attempts with over 10,000 tokens per proof attempt, which raises

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 45741–45753

critical questions about deployment feasibility and resource efficiency. The current implementations lack systematic optimization of the fundamental trade-off between computational cost and performance gains. Moreover, existing analyses of sampling cost typically focus only on the number of sampling passes, neglecting the impact of token usage and iterative refinement steps. This lack of a unified framework limits our understanding of the true computational trade-offs across different scaling strategies.

To systematically analyze these inefficiencies, we take a unified *sampling cost* metric that sums the total token generation costs across all passes and refinement steps. Our analysis reveals that current methods achieve marginal performance gains at disproportionate computational costs: (1) for problems below IMO difficulty level, models can achieve high performance without complex CoT reasoning, suggesting that applying elaborate chain-of-thought approaches to simpler problems may incur unnecessary computational overhead. (2) Parallel sampling can easily generate redundant proof attempts, causing an early performance plateau. Motivated by these insights, we propose a unified framework **EconRL** combining two complementary techniques. First, **dynamic CoT switching** trains models to autonomously activate extended reasoning only for complex problems via preference learning (Rafailov et al., 2024), maintaining comparable accuracy while reducing token usage to 12%. Second, **diverse parallel-scaled RL** employs specialized reasoning heads trained via PPO (Schulman et al., 2017) on difficulty-partitioned data to improve solution diversity and coverage. Integrating iterative refinement further amplifies the benefits of both techniques.

Our main contributions are as follows:

- **Token-level Cost**: We propose evaluating the inference cost of ATP models in terms of the total number of generated tokens, rather than the number of sampling passes.
- **Inference Efficiency Analysis**: We identify substantial token inefficiency in SOTA provers. For instance, 83% miniF2F statements below IMO difficulty level can be solved in Non-CoT mode by leading open-source provers (e.g., DeepSeek-Prover-V2), which requires only about 1/10 of the token budget compared to CoT mode. Furthermore, we observe considerable redundancy across different sampled outputs.
- **EconRL**: We proposed a pipelined framework **EconRL** that stacks two RL tuning stages to improve efficiency. The first stage (**Dynamic CoT Switching**) conducts preference learning to enable autonomous CoT/Non-CoT mode selection, and the second stage (**Diverse Parallel-scaled RL**) introduces specialized reasoning heads trained on difficulty-partitioned data, significantly improving solution diversity and parallel scaling efficiency.
- **Practical Efficiency**: Our **ECONPROVER-GD** matches the performance of the backbone SOTA open-source ATP model while reducing token consumption to merely 12%.

2 Analysis on Test-Time Scaling of ATP models

2.1 Token-Level Sampling Cost Quantification

We formalize the computational cost of different test-time scaling methods for ATP models through *token-level sampling cost*. The sampling cost is calculated as the sum of initial tokens generated in each pass and the tokens generated during all refinement steps. This framework captures the total inference cost by summing all generation costs across passes and refinement steps, providing a unified metric for evaluating cost-performance trade-offs.

2.2 Scaling Efficiency Curves

As shown in Figure 2, we provide a detailed analysis based on our proposed token-level sampling cost and the model's performance on the miniF2F (Zheng et al., 2022) benchmark. We examine how model performance varies under parallel scaling (PS) and sequential scaling (SS). Here, SS denotes increasing the token length by enabling reflective CoT mode, while PS=x indicates using x parallel sampling passes. Our analysis reveals three key insights into the efficiency of different scaling strategies:

Sequential scaling demonstrates lower efficiency compared to parallel scaling For example, when taking 8 passes (PS=8), parallel scaling already achieves better performance (63.1% vs 58.6%) while using fewer tokens (443×8 vs 4,488 tokens) compared to sequential scaling, which enables longer reasoning via reflective CoT. When further increasing the sampling passes to 16, the

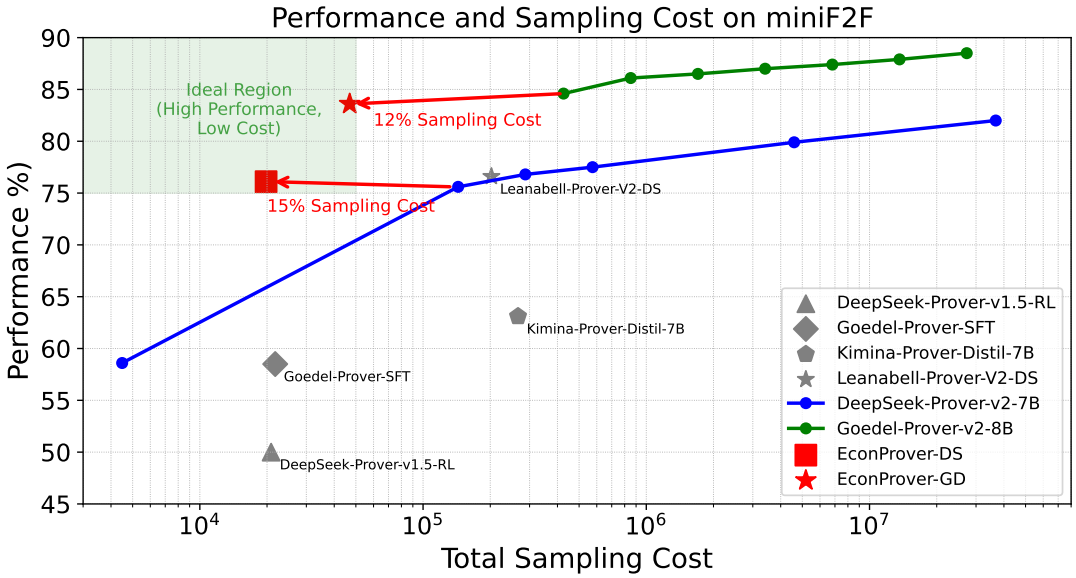

Figure 1: Performance-cost curve of open-source ATP models. The horizontal axis shows the total token-level sampling cost (in log scale), and the vertical axis shows the accuracy on miniF2F. Our **EconProver-GD** achieves comparable performance while requiring only 12% of the sampling cost compared to the base model.

performance can reach 65.6%, another 2.5% increase, while the sampling cost is not significantly higher than sequential scaling.

Diminishing Returns in Parallel Scaling Parallel scaling exhibits clear diminishing returns after increasing passes beyond a certain threshold. While initial increases in passes (from 8 to 32) show substantial gains of 4.9 points, the accuracy improvement becomes increasingly marginal despite exponential growth in computational cost. For instance, doubling passes from 64 to 128 yields only a 1.1% accuracy gain, indicating that simply scaling up the number of parallel samples can quickly face performance plateau.

Hybrid Scaling Benefits Hybrid methods that combine both scaling approaches achieve better efficiency than further increasing parallel passes alone. For example, while increasing PS alone from 32 to 8192 passes yields a 7% gain, combining PS with sequential scaling (PS=32 w/ SS) achieves comparable improvement (7.5%) with only around 4% computational cost than the former (PS=8192). These hybrid methods are simultaneously constrained by the inherent inefficiencies of their components, suggesting that optimizing cost-efficient ATP models requires addressing both the token inefficiency in sequential scaling and the sampling redundancy in parallel scaling simultaneously.

2.3 Analysis on Scaling methods

To systematically investigate the inefficiencies identified in our scaling analysis, we design targeted experiments to examine both scaling methods. For sequential scaling, we analyze its effectiveness across problems of varying difficulty levels, revealing unnecessary token consumption for simpler theorems. For parallel scaling, we study the diversity and redundancy of proof attempts under different sampling sizes.

Sequential Scaling We categorize problems in miniF2F-dev into two difficulty levels: Olympiad-level problems from IMO, AIME, and AMC competitions, and moderate-level problems from MATH algebra and number theory sections. As shown in Figure 3 left part, while Long CoT shows only modest improvements on moderate-level mathematical problems, its effectiveness becomes particularly pronounced on more challenging Olympiad-level problems. The performance gap between Long CoT and baseline approaches widens significantly for these harder problems. This pronounced contrast suggests that extended reasoning through Long CoT primarily benefits challenging proofs that require sophisticated multi-step logical inference, while simpler problems can often be solved effectively with direct formal proof generation.

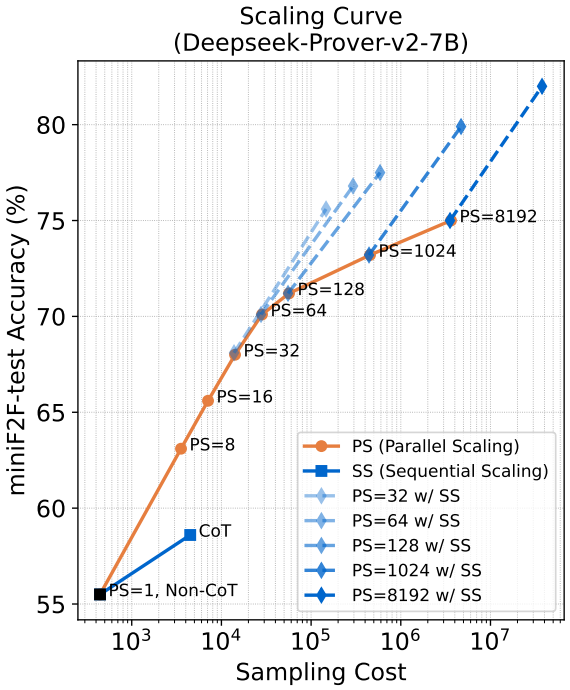

Figure 2: Scaling curve analysis comparing Sequential Scaling (SS), Parallel Scaling (PS), and their combinations (hybrid scaling). The x-axis represents the sampling cost in log scale, and the y-axis shows the miniF2F-test accuracy.

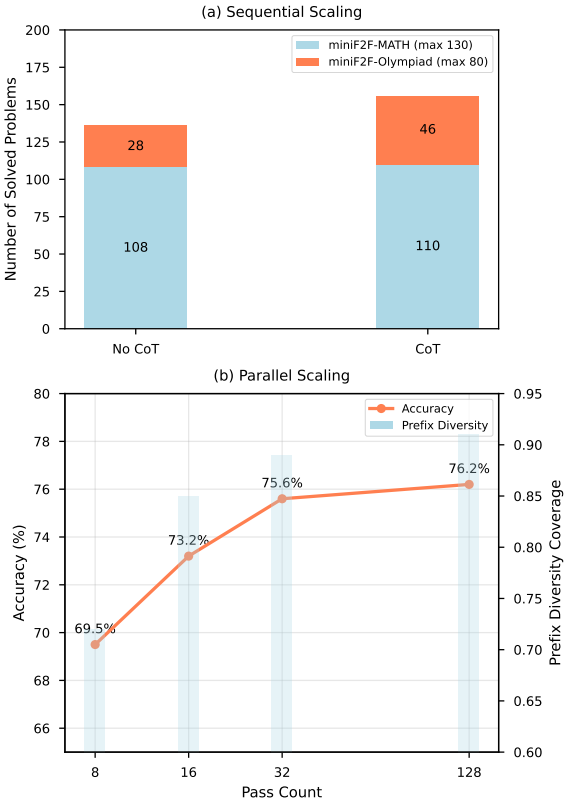

Figure 3: Experiments are conducted on DeepSeek-Prover-V2-7B. **Top** : Sequential scaling improvements are more suitable for solving difficult problems. **Bottom**: Parallel scaling accuracy is highly correlated with prefix diversity in generated content.

Parallel Scaling To analyze the diversity of proof attempts, we measure Prefix Diversity Coverage (PDC). For each attempt, we extract the first 20 tokens after the formal statement and break them into 3-grams. We then track how many distinct 3-grams appear as we increase the number of sampled attempts, relative to the full set of 512 attempts. This gives a coverage value between 0 and 1 that grows monotonically with more samples, reflecting how many unique early proof fragments the model explores. Our analysis reveals that this initial proof prefix largely determines the subsequent proof pattern, with the chosen approach persisting throughout generation. As shown in the right part of Figure 3, prefix diversity coverage strongly correlates with proving performance. While significant gains are observed when scaling from 8 to 32 passes (+6.1% total), further scaling to 128 passes yields minimal returns (+0.6%) despite 4× computational cost. This suggests substantial redundancy in exploration beyond 32 passes, motivating strategies that explicitly encourage diverse prefix generation while maintaining lower pass counts.

3 EconRL

Based on our analysis of current scaling inefficiencies, we propose a unified yet lightweight framework **EconRL** to improve ATP model test-time scaling efficiency from two aspects: **dynamic Chain-of-Thought switching** and **diverse parallel-scaled RL**. The former cuts unnecessary tokens for easy problems, while the latter squeezes more performance out of a small pass budget by encouraging generation diversity. We first introduce each component and then show that their combination yields further gains in the experiments section.

3.1 Dynamic Chain-of-Thought Switching

The inefficiency of uniform CoT application across all problems motivates our development of a dynamic switching mechanism that activates extended reasoning only when necessary. This approach is inspired by recent advances in adaptive CoT triggering for informal reasoning (Lou et al., 2025a; Aggarwal and Welleck, 2025; Pan et al., 2024; Shen et al., 2025). In contrast to those settings, formal

theorem proving provides a crucial advantage: the Lean verifier offers an *exact, noise-free* difficulty signal (proof succeeds or fails), which we exploit to build precise per-problem preference pairs for DPO. To our knowledge, this is the first application of adaptive CoT triggering to Lean-based ATP, and we show in §4 that verifier-grounded preference learning substantially outperforms heuristic triggers based on model confidence.

Problem Complexity Classification We develop a systematic approach to classify problems based on whether they can be solved by Non-CoT mode. Problems are categorized into two classes.

- **Non-CoT Solvable**: problems correctly solved by base model without CoT reasoning.
- **CoT-dependent**: problems that require extended reasoning for successful proof completion.

This classification is performed automatically by running baseline models in the training set and analyzing success patterns. As shown in Figure 3, roughly 83 % of the solved problems of Deepseek-Prover-V2-7B below IMO difficulty level are base-solvable, revealing large potential for computational savings.

Preference Learning We construct preference pairs $(y_{\text{base}}, y_{\text{CoT}})$ for training the switching mechanism with Direct Preference Optimisation (DPO) (Rafailov et al., 2024). The objective is

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(x,y_w,y_l)\sim D}\Big[\log\sigma\Big(-\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} + \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\Big)\Big] \quad (1)$$

with $\beta = 0.1$. The preferred response y_w is the direct proof for base-solvable problems and the full CoT answer for CoT-dependent problems. The preference construction strategy varies by problem types. For base-solvable problems: y_w is the direct proof, y_l is the CoT reasoning (to discourage unnecessary reasoning). For CoT-dependent problems: y_w stands for CoT reasoning, while y_l is the direct proof (to encourage extended thinking). This asymmetric preference design enables the model to learn when to apply extended reasoning while avoiding computational waste on simple problems.

Training Data Construction We curate a balanced training dataset of 15,000 problem-solution pairs from LeanWorkbook (Ying et al., 2025) and Goedel Pset[1]. The dataset composition is 40% base-solvable problems with direct solution preferences and 60% CoT-dependent problems with extended reasoning preferences.

Prompting Strategy Our unified prompt structure, adapted from commonly used prompts in current ATP models to ensure seamless distribution alignment between CoT and Non-CoT modes, enables the model to dynamically determine the appropriate reasoning mode without explicit complexity indicators. During inference, the model learns to assess problem complexity implicitly through attention patterns learned during DPO training. The model develops internal representations that correlate with problem difficulty, enabling automatic mode selection without external complexity signals.

3.2 Diverse Parallel-scaled RL

As discussed in our analysis section, parallel scaling can consistently improve performance by increasing the number of independent proof attempts. However, the marginal gains diminish rapidly at higher pass counts, as many attempts become redundant and repeatedly explore similar solution paths, leading to significant computational overhead for small improvements. To address this inefficiency, we aim to make each attempt more diverse, thereby increasing the efficiency of parallel exploration under a limited sampling budget. Concretely, we learn n specialized heads, implemented as lightweight prefix embeddings, that cooperate to cover the proof space more effectively.

Difficulty-aware Partitioning Our key insight is that difficulty-aware data partitioning enables learning specialized prefix tokens that generate more diverse proof attempts. Specifically, we measure problem difficulty by running the base prover for 32 attempts and recording the success count $c(x)$ for each problem. After sorting problems by $c(x)$, we partition the dataset into n difficulty-based bins $B_1, \ldots, B_n$ from easy to hard. For each bin B_i, we construct a training shard S_i containing 50% problems from B_i and 50% sampled uniformly from other bins, ensuring both specialization and

[1]https://huggingface.co/datasets/Goedel-LM/Goedel-Pset-v1

Dynamic CoT Prompt Template

```
Complete the following Lean 4 code, thinking step by step if the problem
requires careful reasoning:
<problem_statement>
/- The model will autonomously choose:
Option 1:  Direct formal proof generation
Option 2:  Informal reasoning followed by formal proof -/
```

generalization. Each shard is paired with a dedicated prefix (head) P_i that learns difficulty-specific proof strategies. Through extensive experiments shown in §4.4, we found that this difficulty-aware partitioning significantly outperforms naive diversification approaches like random heads, which yield only marginal improvements. Based on empirical validation, we set $n = 8$ heads as the optimal configuration for balancing diversity and computational cost.

Learnable Parallel Scaling via Trainable Prefixes We employ Proximal Policy Optimization (PPO) (Schulman et al., 2017) to train each prefix (head) *independently*, without joint optimization across heads. For each prefix, the training objective is defined as follows. For a given proof attempt, the model receives a reward of 1 if the generated proof is correct, and 0 otherwise. Notably, each prefix is optimized separately, and there is no joint or coordinated update across the eight heads. This independent optimization framework allows each prefix to specialize in different proof strategies, thereby increasing the diversity and coverage of the parallel search process.

Uniform Allocation at Inference During evaluation we distribute the sampling budget evenly across the heads (e.g., Pass@16 $\Rightarrow 16/n$ calls per head). This zero-cost scheduler retains the learned diversity and avoids expensive per-problem head selection.

4 Experiments

We conduct extensive experiments using two SOTA ATP models to evaluate the efficiency improvements of **EconRL** on two popular benchmarks.

4.1 Experimental Setup

Baselines and Models We take **DeepSeek-Prover-V2-7B** and **Goedel-Prover-V2-8B** as the baselines to conduct our experiments. Particularly, **DeepSeek-Prover-V2-7B** (Ren et al., 2025) is built upon DeepSeek-Prover-V1.5-Base and features an extended context length of up to 32K tokens. It leverages Lean 4 integration for formal verification and synthetic data generation via DeepSeek-V3 decomposition. **Goedel-Prover-V2-8B** (Lin et al., 2025b) is built by careful expert iteration and checkpoint weights merging. It also features the capability of progressive proof refinement by the feedback from Lean 4 engine.

Our **ECONPROVER-DS** and **ECONPROVER-GD** are based on **DeepSeek-Prover-V2-7B** and **Goedel-Prover-V2-8B**, trained by **EconRL**. Our **EconRL** (§3) enables the base model by training with the dynamic CoT switching method (§3.1) and subsequently with diverse parallel-scaled RL training (§3.2). For CoT and Non-CoT modes, we adopt the prompt used in the original paper. For iterative refinement, we follow the setting in Goedel-Prover-V2. Detailed settings are shown in Appendix B.

Other Systems **Goedel-Prover-SFT** (Lin et al., 2025b) is obtained by conducting SFT based on DeepSeek-Prover-V1.5. **Kimina-Prover-Preview-Distill-7B** (Wang et al., 2025) combines symbolic reasoning with neural guidance through interactive proof-state pruning. **Leanabell-Prover-V2-DS** (Ji et al., 2025) is developed by further tuning DeepSeek-Prover-V2 with RL feedback from Lean 4. More related work is shown in Appendix A.

Benchmarks We evaluate our approach on two standard theorem proving benchmarks: **miniF2F-test** (Zheng et al., 2022) and **ProofNet-test** (Azerbayev et al., 2023). We follow recent work and use the revised version of miniF2F provided by KiminaProver (Wang et al., 2025), which corrects several errors in the original dataset. MiniF2F consists of 488 problems in Lean, drawn from high-school level competitions such as the AMC, AIME, and the IMO. The benchmark covers core areas of elementary mathematics (e.g., algebra, number theory, and induction) and is split evenly into miniF2F-valid and miniF2F-test, each containing

Model / Setting	miniF2F			ProofNet		
	Cost	Acc (%)	Rel.	Cost	Acc (%)	Rel.
Goedel-Prover-SFT	0.7k×32	58.5	–	0.7k×32	15.6	–
Kimina-Prover-distil-7B	8.3k×32	63.1	–	–	–	–
Leanabell-Prover-V2-DS	6.3k×32	76.6	–	8.9k×32	23.7	–
Deepseek-Prover-V2-7B						
Non-CoT mode	0.4k×32	68.1	1×	0.5k×32	21.5	1×
CoT mode	4.5k×32	75.8	10×	7.0k×32	23.1	10×
ECONPROVER-DS	1.2k×16	76.2	1.5×	2.2k×16	23.1	2×
Goedel -Prover-V2-8B						
Non-CoT mode	0.5k×32	75.8	1×	0.5k×32	24.7	1×
CoT mode	12.3k×32	84.4	25×	15.2k×32	28.5	30×
ECONPROVER-GD	2.9k×16	84.0	3×	3.1k×16	28.0	3×
CoT mode + IR	12.3k+3.7k×2)×32	86.0	40×	–	–	–
ECONPROVER-GD + IR	(2.9k+3.6k×2)×16	86.0	10×	–	–	–

Table 1: Accuracy and efficiency comparison on miniF2F-test and ProofNet-test. Sampling cost is calculated by summing all generation costs across passes and refinement steps. Rel. denotes the relative sampling token cost, expressed as an approximate multiple of the same model's non-CoT baseline.

244 problems. We reserve the test split for evaluation. For ProofNet-test, we follow the data sources and splits used in DeepSeek-Prover-V2 (Ren et al., 2025). In particular, the dataset consists of 371 problems in Lean 3, drawn from a range of popular undergraduate pure mathematics textbooks, covering topics such as real and complex analysis, linear algebra, abstract algebra, and topology. Following DeepSeek-Prover-V2, we use the Lean 4 translation version of the ProofNet test set, which consists of 186 problems. We focus on miniF2F and ProofNet for the main experiments because 7B-scale provers remain near the solvable ceiling on substantially harder benchmarks: on PutnamBench (Tsoukalas et al., 2024), DeepSeek-Prover-V2-7B solves only 10/658 problems even with 1024 samples, making efficiency comparisons uninformative.

4.2 Main Performance and Efficiency

Efficient Performance through Hybrid Approach As shown in Table 1, our experimental results demonstrate the remarkable efficiency of our hybrid approach in automated theorem proving. By combining dynamic CoT switching with diverse parallel-scaled RL, our **ECONPROVER-DS** achieves performance comparable to full CoT methods while significantly reducing computational overhead. Leanabell-Prover-V2-DS, which is also using DeepSeek-Prover-V2 as the base model, achieves slightly higher performance, but the sampling cost increased nearly 50% than its base model. In contrast, our method maintains the high accuracy of CoT-based approaches (achieving 76.2% on miniF2F-test) while requiring only 15% of the token usage compared to standard CoT implementations. This efficiency gain is achieved through intelligent resource allocation, where the system dynamically determines when to employ detailed reasoning steps and optimizes exploration through specialized reasoning heads.

Strong Generalization Across ATP Models The effectiveness of our **EconRL** shows strong generalization across different model architectures and benchmarks. When applied to both DeepSeek-Prover-V2 and Goedel-Prover-V2, our **ECONPROVER-DS** and **ECONPROVER-GD** consistently delivers substantial efficiency improvements while maintaining competitive performance. On miniF2F-test, the approach achieves 76.2% accuracy with DeepSeek-Prover-V2 and 84.0% with Goedel-Prover-V2, demonstrating its adaptability across different foundation models. Similarly, on ProofNet, our method maintains the strong performance of baseline models while significantly reducing computational costs, highlighting the broad applicability of our efficiency optimizations across diverse theorem-proving tasks.

Compatibility with Iterative Refinement Recent advances in ATP systems have demonstrated the effectiveness of iterative refinement (IR) in improving proving performance. As shown in Table 1, Goedel-Prover-V2 with IR achieves 86.0% accuracy, but at the cost of substantially increased token usage (60 times compared to baseline). When combined with IR, our **EconRL** optimizations maintain their effectiveness, reducing the token overhead by 75% while preserving the 86.0% accuracy. This demonstrates that our method can effectively optimize computational costs even in advanced proving frameworks that push the boundaries of performance.

4.3 Ablation on Dynamic CoT Switching

Configuration	Pass@32 (%)	#Avg	CoT Rate (%)
Non-CoT (Baseline)	68.0	443	0.0
Direct Prompt	71.7	1,433	18.1
Log-prob Trigger	73.0	1,684	29.8
Dynamic CoT Switching	75.4	**1,186**	**14.8**
Full CoT (Always)	**75.8**	4,488	100.0

Table 2: Effectiveness of dynamic CoT Switching based on DeepSeek-Prover-V2. *Log-prob Trigger* is a confidence-based heuristic that first produces a Non-CoT proof and invokes CoT only when the average log-probability falls below a threshold $\tau{=}{-}0.125$.

To better understand the effectiveness of our Dynamic CoT Switching approach, we conduct detailed ablation studies using DeepSeek-Prover-V2 as the base model. As shown in Table 2, our method achieves remarkable efficiency while maintaining high performance. Compared to Full CoT, Dynamic CoT Switching achieves 99.7% of the accuracy (75.4% vs 75.8%) while requiring only 15% of the token usage (1,186 vs 4,488.3 average tokens). When compared to the Direct Prompt baseline, our method shows significantly better performance (75.4% vs 71.7%) with better token efficiency. We further compare against a *log-prob*–based confidence trigger, which invokes CoT only when the model's confidence on its Non-CoT attempt is low. This heuristic fires CoT roughly twice as often (29.8% vs 14.8%) yet still underperforms our learned switcher in both Pass@32 (73.0% vs 75.4%) and token cost, indicating that verifier-grounded preference learning provides a stronger difficulty signal than surface-level confidence measures. Moreover, relative to Non-CoT baseline, Dynamic CoT Switching substantially improves accuracy (75.4% vs 68.0%) while keeping the computational overhead at a reasonable level through intelligent CoT rate control (14.8%).

4.4 Ablation on Diverse Parallel-scaled RL

Method	#Heads	Pass@8 (%)	Pass@16 (%)	Diversity Cover@8
Baseline	-	63.1	65.6	74.3
Random	4	65.6	66.8	74.5
	8	66.8	68.9	74.8
	16	66.8	68.9	75.1
Ours	4	67.2	69.3	82.3
	8	**68.9**	**70.5**	84.1
	16	**68.9**	70.1	**84.5**

Table 3: Random vs. our difficulty-aware grouping based on DeepSeek-Prover-V2-8B Non-CoT mode. "Cover@k" represents "Prefix Diversity Coverage@k".

Number of Parallel Heads As shown in Table 3, we first investigate the impact of different numbers of parallel heads ($n \in \{4, 8, 16\}$). With a random grouping strategy, increasing heads from 4 to 8 improves Pass@16 from 66.8% to 68.9%, while further increasing to 16 heads yields no additional gains (68.9%).

Difficulty-aware Data Grouping We then examine the effectiveness of our difficulty-aware grouping strategy compared to random grouping. With the same number of heads ($n = 8$), difficulty-aware grouping significantly outperforms random grouping on both CoT and Non-CoT settings. This improvement is consistent across different head numbers, demonstrating that grouping training data based on problem difficulty helps specialized heads better capture different reasoning patterns.

Results Table 4 demonstrates the effectiveness of our diverse parallel-scaled RL approach across different settings. In the Non-CoT setting, our method significantly improves performance with fewer passes, achieving 70.5% at Pass@16 compared to the baseline's 65.6%. Similarly, with CoT enabled, we observe consistent gains, reaching 75.2% at Pass@16 versus 73.0% baseline. Notably, our approach shows stronger benefits at lower pass counts (Pass@16) compared to higher ones (Pass@32), indicating its effectiveness in maximizing performance when computational resources are constrained. The results also show that simply using random heads provides minimal benefits,

Configuration	Pass@16 (%)	Pass@32 (%)
DeepSeek-Prover-V2		
CoT mode	73.0 ± 0.4	75.6 ± 0.2
+ 8 heads (random)	73.2 ± 0.2	75.8 ± 0.4
+ 8 heads (cluster)	74.8 ± 0.2	76.0 ± 0.2
+ 8 heads (diff-aware)	$\mathbf{75.2 \pm 0.2}$	$\mathbf{76.4 \pm 0.2}$

Table 4: Random vs diverse parallel-scaled RL on CoT mode with larger sampling passes on DeepSeek-Prover-V2 CoT mode. "cluster" groups problems by solution-pattern similarity rather than by difficulty (see Appendix C). All numbers are averaged over two independent sampling seeds; standard deviations across seeds are reported.

highlighting the importance of our difficulty-aware training strategy.

Diversity vs. inference-time routing A natural question is whether the benefit of difficulty-aware partitioning stems from routing problems to "correct" specialized heads at inference. Since we allocate the sampling budget uniformly across heads (§3.2), no such routing occurs. To isolate the role of *diversity* itself, we replace the difficulty partition with a *solution-pattern cluster* partition that groups training problems by the similarity of the first 20 proof tokens produced by the base model (see Appendix C). As shown in Table 4, cluster-based grouping also clearly outperforms both the no-head baseline and random grouping, confirming that the gain comes from inducing distinct proof priors across heads, not from difficulty labels per se. Difficulty-aware partitioning still performs best, likely because problems of different difficulty profiles naturally induce different strategy distributions, but any partitioning that yields semantically distinct specializations recovers most of the gain. We also observe that gains are largest in the economical regime (small pass budgets); once the pass budget is large enough that prefix diversity coverage saturates (Figure 3), the marginal benefit of any partitioning scheme naturally shrinks to 1–2 problems at Pass@32.

5 Conclusion

In this work, we presented a comprehensive approach **EconRL** to improve the test-time scaling efficiency of ATP models through two complementary techniques: dynamic CoT switching and diverse parallel-scaled reinforcement learning. Our dynamic CoT mechanism significantly reduces token overhead by selectively applying extended reasoning, while our parallel RL strategy improves performance with limited budgets through specialized proof heads. Together, our **EconProver** demonstrates that substantial efficiency gains are possible without sacrificing proving capability.

Limitations

Although our current approach is compatible with iterative refinement, we have not yet explored a unified optimization framework that jointly considers iterative refinement, sequential scaling, parallel scaling, and RL training. This limits the extent to which efficiency gains can be systematically realized across all dimensions of the proving process. As future work, we plan to integrate these components into a comprehensive optimization strategy. Such a holistic design has the potential to further reduce inference costs across the entire proving pipeline while preserving or enhancing proving performance, ultimately improving the practicality of advanced ATP systems in real-world applications.

Ethical Consideration

The ethical considerations for our study involve several key aspects: (i) Data sourcing: All datasets employed in our experiments were obtained from publicly available repositories, with strict adherence to their corresponding open-source licenses. (ii) Model usage: Our additional training and evaluation of ATP models, including Deepseek-Prover-V2 and Goedel-Prover-V2, fully comply with their license terms. (iii) Evaluation methodology: We relied solely on automated evaluation pipelines to assess model performance, thereby eliminating the need for human annotators and avoiding potential ethical concerns related to human subject involvement.

References

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

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. 2023. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics.

Chenrui Cao, Liangcheng Song, Zenan Li, Xinyi Le, Xian Zhang, Hui Xue, and Fan Yang. 2025. Reviv-

ing DSP for advanced theorem proving in the era of reasoning models (dsp+). *CoRR*, abs/2506.11487.

Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, Cheng Ren, Jiawei Shen, Wenlei Shi, Tong Sun, He Sun, Jiahui Wang, Siran Wang, Zhihong Wang, Chenrui Wei, Shufa Wei, Yonghui Wu, Yuchen Wu, Yihang Xia, Huajian Xin, Fan Yang, Huaiyuan Ying, Hongyi Yuan, Zheng Yuan, Tianyang Zhan, Chi Zhang, Yue Zhang, Ge Zhang, Tianyun Zhao, Jianqiu Zhao, Yichi Zhou, and Thomas Hanwen Zhu. 2025. Seed-prover: Deep and broad reasoning for automated theorem proving.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Xingguang Ji, Yahui Liu, Qi Wang, Jingyuan Zhang, Yang Yue, Rui Shi, Chenxi Sun, Fuzheng Zhang, Guorui Zhou, and Kun Gai. 2025. Leanabell-prover-v2: Verifier-integrated reasoning for formal theorem proving via reinforcement learning.

Yang Li, Dong Du, Linfeng Song, Chen Li, Weikang Wang, Tao Yang, and Haitao Mi. 2024. Hunyuanprover: A scalable data synthesis framework and guided tree search for automated theorem proving. *CoRR*, abs/2412.20735.

Zhenwen Liang, Linfeng Song, Yang Li, Tao Yang, Feng Zhang, Haitao Mi, and Dong Yu. 2025a. Mps-prover: Advancing stepwise theorem proving by multi-perspective search and data curation. *arXiv preprint arXiv:2505.10962*.

Zhenwen Liang, Linfeng Song, Yang Li, Tao Yang, Feng Zhang, Haitao Mi, and Dong Yu. 2025b. Towards solving more challenging imo problems via decoupled reasoning and proving. *arXiv preprint arXiv:2507.06804*.

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, et al. 2025a. Goedel-prover: A frontier model for open-source automated theorem proving. *arXiv preprint arXiv:2502.07640*.

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, Jiayun Wu, Jiri Gesi, David Acuna, Kaiyu Yang, Hongzhou Lin, Yejin Choi, Danqi Chen, Sanjeev Arora, and Chi Jin. 2025b. Goedel-prover-v2: The strongest open-source theorem prover to date.

Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. 2025a. Adacot: Pareto-optimal adaptive chain-of-thought triggering via reinforcement learning.

Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. 2025b. Adacot: Pareto-optimal adaptive chain-of-thought triggering via reinforcement learning. *arXiv preprint arXiv:2505.11896*.

Leonardo de Moura and Sebastian Ullrich. 2021. The lean 4 theorem prover and programming language. In *International Conference on Automated Deduction*, pages 625–635. Springer.

OpenAI. 2024. Openai o1 system card.

Jiabao Pan, Yan Zhang, Chen Zhang, Zuozhu Liu, Hongwei Wang, and Haizhou Li. 2024. Dynathink: Fast or slow? a dynamic decision-making framework for large language models. *arXiv preprint arXiv:2407.01009*.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model.

ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, et al. 2025. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

Shijie Shang, Ruosi Wan, Yue Peng, Yutong Wu, Xiong hui Chen, Jie Yan, and Xiangyu Zhang. 2025. Stepfun-prover preview: Let's think and verify step by step.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. 2025. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*.

Peiyang Song, Kaiyu Yang, and Anima Anandkumar. 2024. Lean copilot: Large language models as copilots for theorem proving in lean. *arXiv preprint arXiv:2404.12534*.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. 2024. Putnambench: Evaluating neural theorem-provers on the putnam mathematical competition. *Advances in Neural Information Processing Systems*, 37:11545–11569.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, et al. 2025. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*.

Sean Welleck and Rahul Saha. 2023. Llmstep: Llm proofstep suggestions in lean. *arXiv preprint arXiv:2310.18457*.

Makarius Wenzel, Lawrence C Paulson, and Tobias Nipkow. 2008. The isabelle framework. In *International Conference on Theorem Proving in Higher Order Logics*, pages 33–38. Springer.

Zijian Wu, Suozhi Huang, Zhejian Zhou, Huaiyuan Ying, Jiayu Wang, Dahua Lin, and Kai Chen. 2024. Internlm2.5-stepprover: Advancing automated theorem proving via expert iteration on large-scale LEAN problems. *CoRR*, abs/2410.15700.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. 2024a. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data.

Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. 2024b. Deepseek-prover-v1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search.

Ran Xin, Chenguang Xi, Jie Yang, Feng Chen, Hang Wu, Xia Xiao, Yifan Sun, Shen Zheng, and Kai Shen. 2025. Bfs-prover: Scalable best-first tree search for llm-based automatic theorem proving. *CoRR*, abs/2502.03438.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. Leandojo: Theorem proving with retrieval-augmented language models. *Advances in Neural Information Processing Systems*, 36:21573–21612.

Huaiyuan Ying, Zijian Wu, Yihan Geng, Zheng Yuan, Dahua Lin, and Kai Chen. 2025. Lean workbook: A large-scale lean problem set formalized from natural language math problems.

Xueliang Zhao, Lin Zheng, Haige Bo, Changran Hu, Urmish Thakker, and Lingpeng Kong. 2024. Subgoalxl: Subgoal-based expert learning for theorem proving. *ArXiv*, abs/2408.11172.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*.

A Related Work

A.1 Formal Theorem Proving via whole proof generation

Early efforts in learning-based theorem proving for Lean and related systems explored *tactic-level* proof generation combined with search, including **GPT-f** (Polu and Sutskever, 2020), **LeanDojo** (Yang et al., 2023), **LLMStep** (Welleck and Saha, 2023), and **Lean Copilot** (Song et al., 2024). These systems generate one proof step at a time and rely on external proof-search procedures to assemble complete proofs. Inspired by the success of end-to-end reasoning in informal LLMs, recent work in formal theorem proving has instead explored generating entire proofs in one forward pass. In environments like Lean (Moura and Ullrich, 2021), a model's output can be formally verified, avoiding the logically flawed yet fluent arguments often seen in free-form reasoning. Several large formal models such as **DeepSeek-Prover** (Xin et al., 2024a), **Goedel-Prover** (Lin et al., 2025a), demonstrate that whole-proof generation can produce correct Lean proofs that pass the Lean checker. This paradigm retains the global coherence and simplicity of sequence generation while ensuring rigor through verification. Recently, with the rise of long chain-of-thought (CoT) reasoning (OpenAI, 2024; DeepSeek-AI, 2025), ATP models have also started adopting long CoT generation before producing Lean code (Xin et al., 2024b; Wang et al., 2025). While these methods can further improve performance on theorem proving tasks, one challenge in this approach is the length of generated proofs: using explicit chain-of-thought (CoT) reasoning can make proof outputs significantly longer (Ren et al., 2025), inflating token usage. Our work follows this whole-proof direction, further **reducing the token length** by allowing the model to invoke detailed reasoning only when necessary. This yields more compact proofs and directly cuts down token consumption.

A.2 Formal Theorem Proving with Verifier Feedback

Research leveraging verifier feedback in formal theorem proving has evolved along two main directions: stepwise search-based provers that construct proofs incrementally with per-step verification, and whole-proof refinement approaches that generate complete proofs and iteratively improve them based on verifier feedback.

In the first direction, stepwise search-based provers explore a tree of proof states, expanding one step at a time and using the proof assistant to check each step. Systems such as **InternLM2.5-StepProver** (Wu et al., 2024), **HunyuanProver** (Li et al., 2024; Liang et al., 2025a), **DeepSeek-Prover-V1.5** (Xin et al., 2024b), and **BFS-Prover** (Xin et al., 2025) employ best-first or Monte Carlo tree search to try many possible tactic sequences. **DSP+** (Cao et al., 2025) and **DRP** (Liang et al., 2025b) use a powerful general LLM to decompose (Zhao et al., 2024) the problem and then delegate fine-grained steps to a formal prover. This approach improves success rates by iteratively repairing failures, which means if a step is invalid, the model backtracks and tries a different branch – thereby eventually discovering a valid proof if one exists. However, the exhaustive search comes at the cost of **huge computational overhead**: the model must evaluate a large number of partial proofs, making these methods extremely resource-intensive.

The second direction focuses on iterative refinement, where models first generate complete proofs and then iteratively refine them based on verifier feedback. Notable examples include **Seed-Prover** (Chen et al., 2025), **Goedel-Prover-V2** (Lin et al., 2025b) and **StepFun-Prover** (Shang et al., 2025), which use sophisticated refinement strategies to improve initially generated proofs. At the extreme end, **Seed-Prover** (Chen et al., 2025) combines multi-stage reinforcement learning, an agentic search strategy, and extensive test-time exploration to reach *IMO medal-level* performance, fully solving 5 of 6 problems at IMO 2025 and essentially saturating the miniF2F benchmark. This represents a remarkable leap in capability, but it demands enormous computational resources (dozens of refinement iterations). While iterative refinement approaches achieve strong performance, they require substantial computational resources for multiple refinement iterations. Our approach can be effectively combined with these refinement methods while significantly reducing the computational overhead.

A.3 Dynamic Chain-of-Thought

Recent work has explored dynamic Chain-of-Thought (CoT) prompting methods that let an LLM decide on-the-fly whether to engage in multi-step reasoning for a given query, in order to balance accuracy and token cost. For example, **AdaCoT** (Lou et al., 2025b) uses a reinforcement learning policy to trigger CoT only on complex questions, formalizing a Pareto-optimal trade-off between reasoning performance and inference overhead. Similarly, **L1** (Aggarwal and Welleck, 2025) trains an RL-based controller that enforces chain-of-thought length constraints, allowing the model to trade off minimal accuracy loss for significant token savings. Other approaches adopt utility or difficulty-based triggers: **DynaThink** (Pan et al., 2024) dynamically chooses between a "fast" direct answer mode and a "slow" multi-step mode based on the model's confidence in the query, while **DAST** (Shen et al., 2025) adjusts the CoT reasoning depth according to predicted problem difficulty, cutting unnecessary steps without harming performance on challenging queries. All these adaptive CoT frameworks seek to maximize reasoning accuracy on difficult inputs while avoiding verbose reasoning on trivial ones. To our knowledge, however, this paper is the first to apply dynamic CoT triggering in the context of formal theorem proving, where tightly controlling the proof generation length is crucial.

B Experimental setting

We conduct our experiments using state-of-the-art training frameworks and tools. For the dynamic CoT training phase, we utilize LLaMA Factory [2] with 3 training epochs, learning rate of 2e-5, and batch size of 8. We use a cosine learning rate scheduler with warmup and mixed precision training (BF16). For the reinforcement learning phase, we employ OpenRLHF [3] to train our model. The Lean theorem prover environment is set up using the Kimina-Lean server [4], which runs on Lean version 4.9.0. The RL training uses PPO with learning rate 1e-5 and standard PPO hyperparameters. All experiments are conducted on 8 NVIDIA A100-80GB GPUs.

C Alternative Partitioning Schemes for Parallel Heads

To verify that the benefit of diverse parallel-scaled RL comes from inducing distinct proof priors across heads rather than from difficulty labels per se, we compare three partitioning schemes for constructing the n=8 training shards: (i) *random*,

[2] https://github.com/hiyouga/LLaMA-Factory

[3] https://github.com/OpenRLHF/OpenRLHF

[4] https://github.com/project-numina/kimina-lean-server

(ii) *difficulty-aware* (our main setting, based on base-prover success counts at Pass@32), and (iii) *solution-pattern cluster*. For the cluster scheme, we first sample one proof attempt per training problem from the base prover and compute a similarity score between proofs using 3-gram overlap over the first 20 tokens after the formal statement. We then run agglomerative clustering with cosine similarity to obtain 8 clusters, and treat each cluster as a partition B_i, following the same shard-construction protocol as the difficulty-aware case (50% problems from B_i, 50% sampled from other clusters).

Table 4 reports results averaged over two independent sampling seeds. Both difficulty-aware and cluster-based partitioning clearly outperform random grouping and the no-head baseline at Pass@16, where diversity has the largest impact. Variance across seeds is small and the relative ordering is stable, showing that the improvements are robust. Difficulty-aware partitioning remains the best, likely because problems of different difficulty profiles naturally induce different strategy distributions, providing a slightly stronger specialization signal than prefix-based clustering. At larger pass budgets (Pass@32) all methods converge, consistent with the saturation of prefix diversity coverage observed in Figure 3.

Robustness of the Difficulty Labels. We further checked that the difficulty partitioning itself is not sensitive to the sampling policy used to obtain the labels. Re-labelling problems with Pass@16 sampling instead of Pass@32 yields nearly identical partitions and comparable downstream accuracy (75.0$\pm$0.2% at Pass@16, 76.2$\pm$0.4% at Pass@32), confirming that the Lean verifier signal provides a stable difficulty ranking even under modest variance in sampling budget.

D Robustness of Prefix Diversity Coverage

Our main paper reports Prefix Diversity Coverage (PDC) based on 3-grams over the first 20 tokens after the formal statement. To check that this choice does not drive the qualitative conclusions, we recomputed PDC with 4-grams over the first 40 tokens. Absolute coverage decreases slightly under this stricter measure, but the overall trend is preserved: PDC rises sharply from Pass@8 to Pass@32 (approximately 62%$\rightarrow$78%) and grows only marginally from Pass@32 to Pass@128 (78%$\rightarrow$81%). The sharp saturation around 32 passes – and its tight correlation with Pass@k accuracy gains – remains the same. Because prompts used by SOTA provers are highly standardized (we use the official CoT prompt of DeepSeek-Prover-V2), prefix-level n-grams are not confounded by stylistic or templating noise, and both 3-gram and 4-gram variants of PDC yield the same conclusion: early-prefix diversity is the dominant bottleneck of parallel scaling.