Parallel-R1: Towards Parallel Thinking via Reinforcement Learning

Tong Zheng[1,2]*, Hongming Zhang[1], Wenhao Yu[1], Xiaoyang Wang[1], He Xing, Runpeng Dai[1,3], Rui Liu[2], Huiwen Bao[4], Chengsong Huang[5], Heng Huang[2], Dong Yu[1]

[1]Tencent AI Lab Seattle, [2]University of Maryland, College Park,

[3]University of North Carolina at Chapel Hill,[4]City University of Hong Kong

[5]Washington University in St. Louis

Abstract

Parallel thinking has emerged as a novel approach for enhancing the reasoning capabilities of large language models (LLMs) by exploring multiple reasoning paths concurrently. However, activating such capabilities through training remains challenging, as existing methods predominantly rely on supervised fine-tuning (SFT) over synthetic data, which encourages teacher-forced imitation rather than exploration and generalization. Different from them, we propose **Parallel-R1**, the first reinforcement learning (RL) framework that enables parallel thinking behaviors for complex real-world reasoning tasks. Our framework employs a progressive curriculum that explicitly addresses the cold-start problem in training parallel thinking with RL. We first use SFT on prompt-generated trajectories from easier tasks to instill the parallel thinking ability, then transition to RL to explore and generalize this skill on harder problems. Experiments on various math benchmarks, including MATH, AMC23, and AIME, show that Parallel-R1 successfully instills parallel thinking, leading to 8.4% accuracy improvements over the sequential thinking model trained directly on challenging tasks with RL. Further analysis reveals a clear shift in the model's thinking behavior: at an early stage, it uses parallel thinking as an exploration strategy, while in a later stage, it uses the same capability for multi-perspective verification. Most significantly, we validate parallel thinking as a **mid-training exploration scaffold**, where this temporary exploratory phase unlocks a higher performance ceiling after RL, yielding a **42.9%** improvement over the baseline.

1 Introduction

Google's Gemini recently credited its success at the International Mathematical Olympiad in part to a new capability: parallel thinking [Luong and Lockhart, 2025]. This approach, as exemplified by Figure 1 (top), involves jointly conducting both parallel and sequential thinking. This success highlights the value of parallel thinking as more than a technical trick. Indeed, cognitive science suggests that humans often engage in such thinking, considering multiple possibilities simultaneously before synthesizing them into coherent conclusions. This process encourages divergent thought, prevents premature "lock-in" to a single, potentially suboptimal solution, and facilitates structured, deliberate reasoning [Clark, 1989, Jackendoff, 2011]. Inspired by these, we investigate how to effectively instill parallel thinking in large language models (LLMs).

*This work was done during Tong Zheng's internship at Tencent AI Lab Seattle.

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

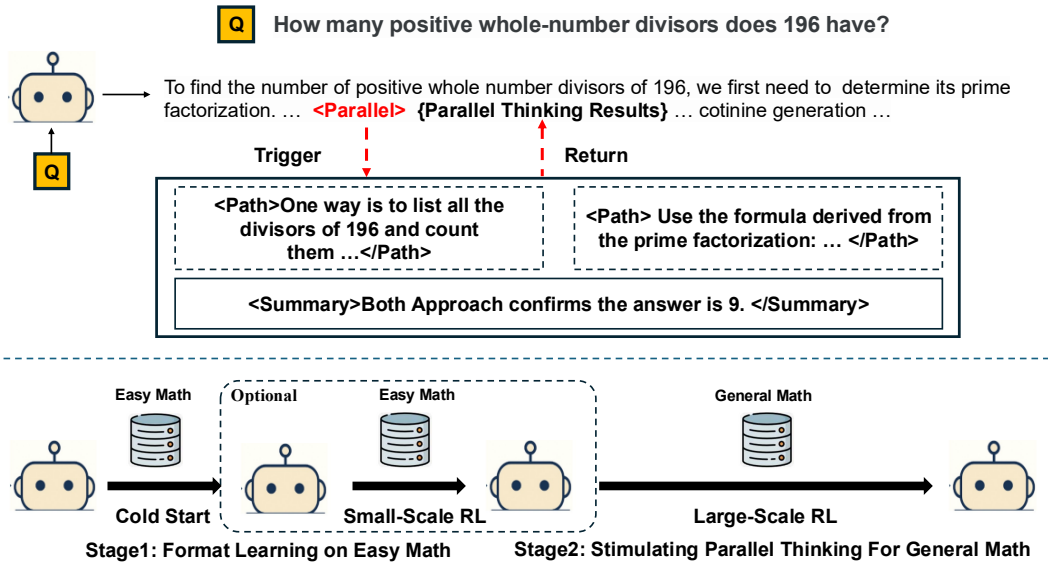

Figure 1: An overview of the proposed framework. (Top) During inference, the model generates in a standard auto-regressive fashion until it emits a special `<Parallel>` tag. At that point, it spawns multiple threads to explore different solution paths or perspectives, then summarizes their outputs. These contents are merged back into the main context, and generation continues. This cycle may repeat several times before the model arrives at the final answer. (Bottom) Parallel thinking ability is obtained by a progressive multi-stage training approach. Intuitively, the approach first equips the model with parallel thinking ability on easy math problems and then progressively extends it to more general and difficult problems through reinforcement learning.

Despite its potential, the question of how to activate parallel thinking remains open. While test-time strategies [Yao et al., 2023, Wang et al., 2022, Brown et al., 2024, Zhang et al., 2024, Hsu et al., 2025, Rodionov et al., 2025, Fu et al., 2025] can elicit such behavior at the cost of high inference overhead, there is a growing interest in permanently instilling this capability through training. However, current training-based approaches fall short of this goal. Methods based on supervised fine-tuning (SFT) [Yang et al., 2025b, Macfarlane et al., 2025, Chen et al., 2025a], for instance, essentially perform behavioral cloning on pre-generated reasoning trajectories. This approach often relies on complex and costly data pipelines to synthesize high-quality parallel thinking data, leading to superficial pattern matching rather than the acquisition of a deep, intrinsic reasoning skill. Consequently, while models can replicate known patterns, their ability to generalize the underlying parallel thinking strategy is severely limited.

In contrast, reinforcement learning (RL) offers a more scalable approach to activating the parallel thinking ability of LLMs since we could let the model explore and learn such behaviors in the wild. However, applying RL to teach models to conduct parallel thinking is not trivial. Since the current LLMs have not seen parallel thinking behavior during the pre-training and sft, they cannot generate such trajectories during explorations for the model to learn from. Thus, the cold-start training becomes crucial. The goal of this stage is to teach the model basic formats without harming it too much, which requires a small-scale, high-quality dataset. However, the fact is that high-quality parallel thinking data for complex, real-world problems is extremely rare in natural text and difficult to synthesize. This explains why successful applications of RL for parallel thinking have been confined to narrow, synthetic domains, such as the CountDown task [Pan et al., 2025]. Additionally, the best reward function for RL remains unclear. If we only use the final correctness as the reward, the model might take shortcuts to forget the complex but better parallel thinking strategy. On the other hand, if we force the model to use a thinking strategy, the model might learn to use parallel thinking in unnecessary scenarios. Lastly, the strategic role and underlying mechanisms of parallel thinking in LLMs are largely a black box. Even if a model acquires this ability, critical questions remain unanswered. For instance, how does the model's strategy evolve throughout training? Without understanding this dynamic, it's impossible to fully unlock the potential of parallel thinking technology.

To address these challenges, we present the **first reinforcement learning framework** designed to help models to learn parallel thinking behavior via exploration on general mathematical reasoning

tasks. First, to resolve the critical cold-start problem, we propose a progressive curriculum. As shown in Figure 1, it begins with supervised fine-tuning on simpler problems, for which we find high-quality parallel thinking data can be generated easily via simple prompting (see Table 1). This initial stage, using our created *Parallel-GSM8K* dataset, effectively teaches the model the basic format of parallel thinking before it transitions to reinforcement learning on more difficult tasks to explore and generalize this new ability. Second, we tackle the critical challenge of reward design by exploring how to balance final accuracy with the desired parallel thinking structure. We propose and investigate multiple reward schemes. Our key finding is an effective alternating reward strategy, which switches between an outcome-based (accuracy) reward and a reward that encourages parallel thinking behaviors within fixed windows, e.g., every 10 steps. We show this approach achieves a superior balance between high performance and consistent utilization of parallel thinking compared to using a single reward type alone. Lastly, to open the "black box" of its strategic role, we conduct a detailed analysis of the model's learned behavior throughout the training process. Our analysis reveals a clear strategic evolution: the model initially leverages parallel paths for computational exploration to discover potential solutions, but as it gains proficiency, its strategy shifts towards using them for multi-perspective verification to confirm the final answer. This finding provides the first empirical evidence of how an LLM's reasoning strategy with parallel thinking evolves, offering crucial insights into the underlying mechanisms that drive its effectiveness. Based on this, we further conceptualize and empirically validate the idea of using parallel thinking as a mid-training exploration scaffold—a temporary exploratory phase that unlocks a higher performance ceiling, notably achieving a peak accuracy of **25.6%** on the challenging AIME25 benchmark. We investigate these contributions across both causal and structured model variants to provide robust insights into architectural design.

In all, our core contributions can be summarized as follows:

- We present the **first RL framework** to learn parallel thinking from scratch on general mathematical reasoning tasks, enabled by our **progressive curriculum** and dedicated **Reward Design**.
- We provide a deep analysis of the learning dynamics, revealing that the model's strategy evolves from **exploration to verification**. We further identify and empirically validate the concept of parallel thinking as a **mid-training exploration scaffold**.
- We provide comprehensive empirical validation, including a comparison of **causal and structured model variants**. Our approach yields consistent gains across multiple benchmarks, and ablations offer practical insights into reward and architectural design.

2 Learning Parallel Thinking via Reinforcement Learning

2.1 Overview

Previous methods for training parallel thinking, such as those in [Yang et al., 2025b, Macfarlane et al., 2025, Chen et al., 2025a], primarily rely on SFT, a paradigm that suffers from several key limitations. By its nature, SFT's success is entirely dependent on the quality of pre-generated training data. This creates a critical dependency on complex and costly data pipelines, especially when generating data for final, challenging problems. Furthermore, this approach constrains the model to merely mimicking known patterns, which hinders the acquisition of a deep, generalizable reasoning skill. To overcome these limitations, we introduce a reinforcement learning (RL) framework.

The key insight of our approach is to bypass the need for the complex data pipelines often considered essential for generating training data on final challenging problems [Yang et al., 2025b, Macfarlane et al., 2025]. Instead, we generate high-quality 'cold-start' data by using simple tasks, and then leverage this data to enable the model to learn parallel thinking on much harder problems via reinforcement learning. We then explore two distinct settings for learning parallel thinking via RL: without architectural modifications and with architectural modifications. Specifically, the latter involves modifying the model's self-attention mask and position ids to prevent cross-attention between parallel reasoning paths, thereby enforcing their structural independence.

In the subsequent sections, we first define our parallel thinking behaviors and their inference workflow. We then describe our data pipeline for generating high-quality training data. Finally, we present our RL training recipes for both settings.

Table 1: Comparison of Parallel-Thinking Data Quality Generated by DeepSeek-R1-0528-Qwen-3-8B on DAPO and GSM8K under identical prompts and sampling settings. The results show that, with simple prompting, state-of-the-art models still struggle to produce concise, high-quality parallel reasoning traces for challenging mathematics problems.

Data	# Samples	Parallel Thinking Format (%)
GSM8K	7472	**83.7**
DAPO	17916	0.0

2.2 Formulation of Parallel Thinking Behaviors

Intuitively, in human problem-solving, we often encounter moments of confusion or uncertainty, which are referred to as "critical steps" within a reasoning chain. At these points, engaging in parallel thinking allows us to explore multiple solution paths simultaneously and converge on a higher-quality conclusion. Inspired by human problem-solving patterns, we formalize LLM's parallel thinking in two stages:

1. **Exploration**: When the model detects a critical step, it temporarily suspends the main chain and launches a multi-thread search, generating N independent trajectories simultaneously.
2. **Summary**: After exploration, the model aggregates the outcomes, distills key insights, and resolves conflicts to arrive at the most promising conclusion. It then automatically resumes the main reasoning chain with the summarized conclusion.

We allow the model to repeat these two phases whenever needed during the reasoning process. We illustrate this process in Figure 1 (Top). To implement this behavior, we introduce three control tags, `<Parallel>...</Parallel>`, `<Path>...</Path>`, and `<Summary>...</Summary>`, which correspond to the exploration phase, the isolation of reasoning threads, and the summary of the parallel thinking, respectively. With these tags, we can define the workflow at the inference phase as follows:

Workflow at Inference Phase At inference time, our model dynamically executes the parallel thinking behaviors as follows: It first conducts auto-regressive generation in the main reasoning process. Whenever it predicts a `<Parallel>` token, it pauses the main reasoning chain and concurrently expands multiple reasoning threads within separate `<Path>...</Path>` blocks. After generating all parallel threads, the model automatically aggregates their outputs into a concise `<Summary>...</Summary>` block, integrating insights from diverse perspectives. Finally, all contexts of parallel thinking are used to resume and complete the main reasoning path. Such adaptive and dynamic parallel inference effectively leverages parallelism.

2.3 The Simple and Scalable Data Pipeline for Parallel Thinking

Collecting high-quality parallel thinking data is a significant challenge. Even though humans think in the parallel fashion, they will summarize and only say/write the summarization. Thus, such data is extremely rare in the natural distribution. Existing approaches, such as the one described in [Yang et al., 2025b], try to solve this by leveraging the inherent parallelism of long CoT reasoning chains. However, these methods rely on complex, multi-stage data pipelines that, while avoiding costly human annotations, are computationally intensive and fundamentally limited in their scalability.

Our approach is based on a key finding from our preliminary experiments. We found that while a simple prompting approach struggles to generate high-quality parallel-thinking data for complex problems like DAPO, it proves highly effective for simpler tasks like GSM8K. The data in Table 1 supports this finding. Based on this discovery, we propose a simple and scalable data pipeline that uses detailed zero-shot prompts to construct a large, high-quality corpus for these easier problems.

As the structured model variant (described in Section 2.5) utilizes architectural modifications like path-window attention masks, it requires strict format adherence for successful training. Therefore, to ensure the quality and alignment of this corpus, we perform an additional filtering step, a Parallel Thinking Format Check, which is implemented by Algorithm 1. Crucially, we make the strategic choice to use this 'cold-start' data not to teach the model how to solve the final target tasks, but specifically to teach it the format of parallel thinking. This initial stage allows us to transition from a data-intensive approach to a more efficient reinforcement learning framework that can learn to elicit and strengthen parallel thinking behaviors from the ground up.

2.4 Eliciting Parallel Thinking via Reinforcement Learning in Causal Models

Unlike prior approaches that use complex and costly data pipelines, we design a simple and scalable data pipeline to efficiently generate a large, high-quality parallel thinking dataset on easy math problems. This dataset serves as a crucial cold start to teach the model the correct format for parallel thinking. Our key idea is to use an RL framework to generalize this format and ability from simple problems to more difficult mathematical tasks. In this section, we explore strategies to elicit this parallel thinking behavior without modifying the model's architecture.

2.4.1 Reinforcement Learning Algorithms

We use Group Relative Policy Optimization (GRPO) [Shao et al., 2024] as our reinforcement learning algorithm. Our model's rollout process follows a multi-turn interactive framework where the LLM alternates between autoregressive generation, parallel exploration, and summarization. The concrete process follows the same procedure as described in Section 2.2.

2.4.2 The Training Recipe and Reward Modeling

The overall training recipe consists of three stages: 1) Cold-Start Stage; 2) RL on Easy Math, and 3) RL on General Math.

Cold-Start Stage We construct and collect a small set of parallel-thinking format examples to fine-tune the initial RL actor using the approach in Section 2.3. Specifically, we use a distilled Qwen3-8B model (i.e., DeepSeek-R1-0528-Qwen-3-8B) to produce high-quality parallel-thinking outputs, extracting only non-thinking parts (final short CoT) as gold annotations.

We select the GSM8K training set, which consists of approximately 7k samples, as the seed dataset. We call the resulting cold-start dataset *Parallel-GSM8K*. This cold-start training is used to teach model the basic format of parallel thinking.

RL on Easy Math After the cold start with SFT, the model already possesses the basic ability to generate the tags for parallel thinking, but the behavior is not stable since this special token has never appeared in the pre-training. To address this issue, we further perform small-scale reinforcement learning to enhance the format learning. In this stage, we use the same question set as the cold-start data and use GRPO for our RL training. To ensure parallel ratio and accuracy, the final reward format in this stage is: $R_{final} = R_{\langle \text{Parallel} \rangle} \times R_{\text{acc}}$. Here, the Accuracy Reward (R_{acc}) evaluates the correctness of the final response, while the Parallel Reward ($R_{\langle \text{Parallel} \rangle}$) incentivizes the model to use parallel reasoning paths. This reward structure is designed to be binary and strict: a positive reward of +1 is given only if the generated output contains at least one parallel thinking unit **AND** the final answer is correct. Otherwise, the model receives a penalty of -1.

RL on General Math After the initial training, the model can stably generate control tags and produce outputs in the correct parallel thinking format if needed, but it still struggles with more challenging mathematical tasks. To address this, we apply reinforcement learning to general math datasets, thereby generalizing the model's parallel thinking ability beyond simple cases.

Specifically, we use the same GRPO algorithm introduced in Section 2.4.1 with accuracy reward (R_{acc}) as our sole reward. This is because the primary goal of this stage is to improve task performance. For the seed problems, we choose the widely used DAPO dataset [Yu et al., 2025]. Finally, the models produced by this stage are our *Parallel-Seen* variants.

2.5 Eliciting Parallel Thinking via Reinforcement Learning in Structure Models

In the previous section, we explored an RL framework that trains models to use parallel thinking without modifying their underlying architecture. However, this approach, which we call Parallel-Seen, does not explicitly isolate reasoning paths. As a result, hidden representations from one path can inadvertently leak into others, and gradients across paths can interfere with each other during training.

To explore an alternative solution, we introduce a structured variant of our framework, Parallel-Unseen. This model incorporates explicit inductive biases into the attention mechanism to enforce

path isolation. Specifically, inspired by prior work [Yang et al., 2025b], we design path-window masking and multiverse position encodings to achieve the goal.

2.5.1 Structured Attention Mechanism

We incorporate these inductive biases directly into the attention layer, as shown in Figure 2.

- Path-window masking restricts each token within a `<Path>` block to attend only to tokens from the same path and the shared context. This prevents cross-path information leakage.
- Multiverse position encodings assign a disjoint set of position indices to each path, ensuring that the positional embedding space does not overlap.

Together, these constraints enforce explicit isolation among reasoning threads while preserving visibility from the shared `<Summary>` block, which is essential for integrating insights across paths.

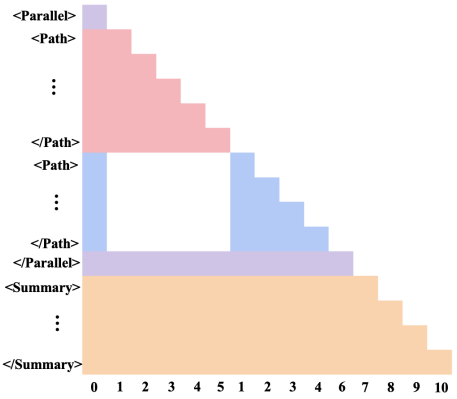

Figure 2: Illustration of the structured attention mask and position IDs, where different paths and the summary block have distinct visibility regions. Blank regions indicate tokens that cannot attend to each other, while colored regions indicate tokens that can.

2.5.2 The Training Recipe and Reward Modeling

In preliminary experiments, we find that directly applying the progressive training recipe from Parallel-R1-Seen to the structured variant proves ineffective. We attribute this to the poor generalization of attention masks from easy to hard math [Yang et al., 2025c]. To address this limitation, we remove the stage one RL and redesign the reward schedule and evaluate two alternative schemes.

(S1) Accuracy-only. We optimize solely for task correctness.

(S2) Alternating accuracy and parallel. In this scheme, we alternate between two different rewards within fixed windows of W=10 steps. For 80% of the steps, we use a standard accuracy-only reward (R_{acc}). For the remaining 20% of the steps, we use a tiered reward system to provide a nuanced incentive for parallel thinking: 1) +1.2: If the generated output contains at least one parallel thinking unit AND the final answer is correct; 2)+1.0: If the generated output does not contain a parallel thinking unit AND the final answer is correct; 3) -1.0: For all other cases, including incorrect answers.

This schedule reintroduces a calibrated incentive for parallel usage without letting it dominate training. Together, these reward designs equip Parallel-R1-Unseen with parallel thinking behaviors.

3 Experiments

3.1 Experimental Setups

Model. We use Qwen-3-4B-Base [Yang et al., 2025a] as our backbone, the latest state-of-the-art open-source model at this scale, offering an ideal balance between performance and efficiency.

Evaluation. We measure our models on four standard mathematical reasoning benchmarks, including AIME'24, AIME'25, AMC'23, and MATH [Hendrycks et al., 2021]. On the MATH dataset, we generate one response per question using a sampling temperature of $T = 1.0$. For the remaining three datasets, we sample 16 independent responses per question at the same temperature and report the average accuracy (i.e., mean@16) to reduce randomness, which is consistent with settings in Wang et al. [2025c]. We additionally report pass@16 to show the upper bound of our approach.

Training Details. Our codebase is adapted from VERL [Sheng et al., 2024], where we primarily follow its official training recipe without any hyperparameter tuning. In the cold start stage, we perform SFT on our curated Parallel-GSM8K, using a batch size of 128, a learning rate of 1e-5, a

Table 2: Performance comparison on mathematical reasoning benchmarks for the Qwen-3-4B-Base model trained under different parallel thinking configurations. We report Mean@16 and Pass@16 for AIME25, AIME24, and AMC23, while MATH is evaluated with Mean@1.

Method	**# Parallel**	**AIME25**		**AIME24**		**AMC23**		**MATH**	**Avg.**
		Mean@16	Pass@16	Mean@16	Pass@16	Mean@16	Pass@16	Mean@1	
Qwen3-4B-Base	0.0	1.3	10.2	2.9	16.5	8.1	51.2	13.9	6.6
SFT + Parallel									
Parallel-SFT-Seen	95.6	8.0	29.8	10.6	26.4	48.9	79.2	76.6	36.0
Parallel-SFT-Unseen	95.6	5.2	20.9	8.5	26.7	41.7	80.1	71.5	31.7
RL Approach									
GRPO (DAPO)	0.0	14.8	32.4	18.5	30.6	63.6	85.1	83.5	45.1
+ RL on GSM8K	0.0	13.3	26.3	18.8	34.9	66.4	82.2	82.6	45.3
Parallel-R1-Seen	27.3	**19.2**	38.9	**19.4**	**37.1**	**70.5**	85.0	**86.7**	**48.9**
Parallel-R1-Unseen (S1)	13.6	17.7	37.8	18.3	33.2	69.7	88.9	82.6	47.1
Parallel-R1-Unseen (S2)	**63.0**	19.0	**42.2**	16.3	31.8	67.5	**91.5**	84.5	46.8

Table 3: Ablation Study on Reward Modeling for the PARALLEL-R1-UNSEEN Model.

Training Configuration	**Parallel Ratio**	**AIME 25**	**AIME 24**	**AMC 23**	**MATH**
Accuracy	13.6	17.7	**18.3**	**69.7**	82.6
Parallel	**80.3**	17.7	15.2	59.4	81.7
Alternating Acc./Parallel	63.0	**19.0**	16.3	67.5	**84.5**

weight decay of 0.01, and a warm-up step ratio of 0.1 with the cosine learning-rate schedule, resulting in 58/230 gradient update steps for Parallel-SFT-Seen and Parallel-SFT-Uneen, respectively. For Stage 1, we optionally perform RL on GSM8K for five epochs, using a batch size of 1024, 5 rollouts, and a learning rate of 1e-6 without warm-up or learning rate scheduling, resulting in 35 gradient update steps. For Stage 2, we perform RL on the DAPO training set for 300 gradient update steps, using a batch size of 512, a rollout of 8, and a learning rate of 1e-6 without warm-up or learning rate scheduling.

3.2 Main Results

Table 2 presents the results across four benchmarks: AIME25, AIME24, AMC23, and MATH. We compare our method against two baselines: 1) RL with GRPO algorithm directly on the DAPO training set, and 2) RL with GRPO in two stages: first trained on the GSM8K data, then further trained with RL on the DAPO training set. The second baseline is included to ensure fair comparison.

Our progressive `Parallel-R1` framework proved to be the most effective approach, consistently outperforming all baselines as shown in Table 2. The top-performing causal variant, `Parallel-R1-Seen`, achieved the highest average score of 48.9. This success stems from a curriculum designed to overcome the limitations of simpler methods. For instance, while SFT provides a substantial foundational improvement (e.g., 31.7 for `Parallel-SFT-Unseen` vs. 4.6 for the base model), it is insufficient for advanced reasoning and falls considerably short of the standard GRPO baseline's score of 45.1. Besides, we found that a naive additional RL on easier data offers only a marginal benefit on average (45.3 vs. 45.1), validating our strategy of using cold start for targeted format and behavior learning.

Our results also reveal key design trade-offs. The superior performance of the Seen model compared to its structured counterparts suggests that explicit architectural modifications can be detrimental to RL training. Furthermore, the comparison between reward schedules for `Parallel-R1-Unseen (S1)` and `(S2)` highlights that reward design is essential for effectively managing the trade-off between the parallel ratio and overall performance. We provide detailed analysis in Section 3.3.1.

3.3 Analysis

3.3.1 Ablation Studies on Reward Modeling: How to Effectively Stimulate Parallel Thinking

In our work, a key question is "how to effectively stimulate parallel thinking behavior." To answer this, we test several reward modeling strategies, including direct accuracy, direct parallel, and an alternating approach. We present the results in Table 3. First, we can see the "Accuracy" configuration, which optimizes solely for problem correctness, yields the highest performance on two out of four benchmarks, particularly on the AMC dataset (69.7). However, this approach yields a very low

parallel ratio of 13.6. In contrast, the "Parallel" configuration, which directly rewards the generation of parallel structures, achieves a high parallel ratio of 80.3. However, this focused optimization leads to a significant performance drop across most benchmarks. With our "Alternating Acc./Parallel" strategy, which periodically switches between rewarding accuracy and parallel structures, provides a superior balance. We also provide more ablation studies on the effect of training stages and parallel thinking prompts in Appendix D.

3.4 Evolution of Parallel Thinking Behavior During RL Training

To better understand how the model's strategy evolves, we analyzed the positional dynamics of the `<Parallel>` block throughout the RL training. We measured the relative position of each block by dividing its starting token index by the total sequence length of the solution. The training dynamics in Figure 3 show a clear and consistent trend: the average relative position of the `<Parallel>` block steadily increases as RL training progresses, indicating a strategic shift from applying this feature early in the reasoning chain toward the very end.

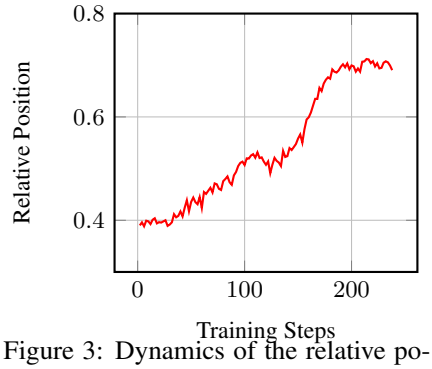

Figure 3: Dynamics of the relative position of the `<Parallel>` block during RL training. The increasing trend indicates the model learns to apply parallel thinking later in the reasoning process.

We interpret this positional shift as the model adopting a more conservative strategy to maximize its reward, a behavior shaped directly by the final-answer-dominated reward design. In the early stages of training, when the model's reasoning ability is weak, using parallel paths for **computational exploration** is a necessary, high-variance strategy to discover a potential solution. However, as the model's core reasoning ability improves, such early-stage exploration becomes a liability that could introduce errors and jeopardize the final reward.

Consequently, the model learns a more risk-averse strategy to secure a correct answer. It first derives a solution using a single, high-confidence reasoning path. Only after a potential answer is found, it deploys the `<Parallel>` block for **multi-perspective verification**. This late-stage use of parallel thinking confirms the result without risking the integrity of the primary solution path, thus maximizing the probability of receiving a positive reward. This learned behavior aligns with our broader finding of a tension between final-answer optimization and the preservation of diverse reasoning structures.

To further illustrate this behavioral evolution, we present two representative case studies below (Figure 5 and 6). The first case, from an early-stage model, demonstrates the use of parallel thinking for exploration. The second, from the late-stage model, exemplifies the learned, verification-oriented strategy.

3.5 Extra Bonus: Parallel Thinking as a Mid-Training Exploration Strategy for RL Training

In this section, we investigate the hypothesis that **parallel thinking itself can serve as an effective structured exploration mechanism to improve RL training.** A fundamental challenge in RL is ensuring the model sufficiently explores the policy space to avoid converging to local optima. We posit that by compelling the model to generate multiple, parallel thought blocks at specific reasoning steps, we introduce a strong inductive bias that forces a more structured and diverse exploration, guiding the model toward more robust policy spaces.

To empirically validate this hypothesis, we designed a two-stage training curriculum, with its dynamics and results presented in Figure 4.

- **Stage-1 (Exploration Phase, steps 0-200)**: The primary goal of this initial phase is to maximize exploration. In this stage, we follow the training approach of our Parallel-R1-Unseen (S2), which explicitly incentivizes the use of the parallel thinking structure by applying an alternating ACC/PAR reward. As shown by the green dashed line in Figure 4, this successfully maintains a high parallel ratio, forcing the model to explore a wide breadth of reasoning paths constantly.
- **Stage-2 (Exploitation Phase, after 200 steps)**: At the 200-step mark, we change the focus from exploration to exploitation. The training objective is then switched to optimize for accuracy alone,

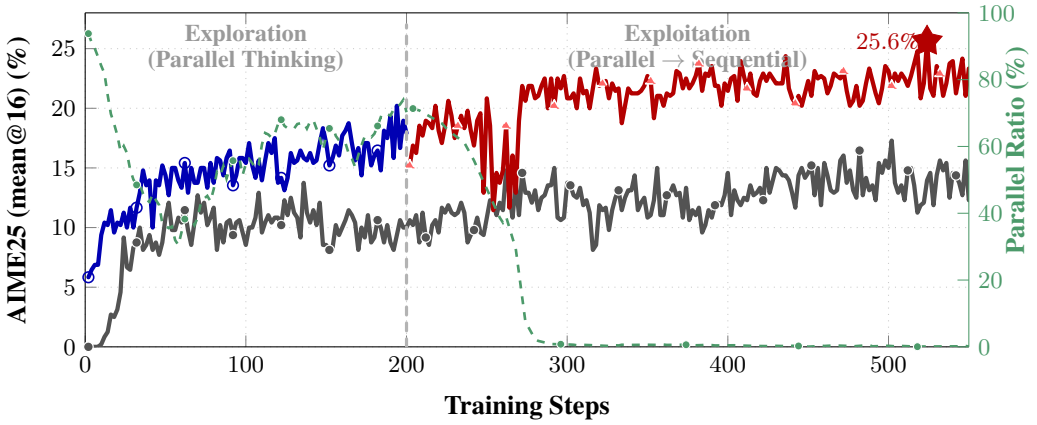

Figure 4: Two-stage training with parallel reasoning as a mid-training exploration scaffold. Left axis plots AIME25 accuracy for Baseline (gray), Stage-1 (blue), and Stage-2 (red); right axis shows the proportion of outputs using the explicit parallel thinking structure. Stage-1 (0–200 steps; vertical dashed line) alternates ACC/PAR rewards to promote exploration, while Stage-2 continues GRPO with an accuracy reward only and is plotted after a +200-step shift to align the timeline. As training transitions from parallel to more sequential reasoning, the parallel ratio decreases yet accuracy continues to improve, peaking at 25.6%, which exceeds single-thread model trained via GRPO.

allowing the model to refine and exploit the effective strategies discovered during the exploration phase.

The experimental results provide evidence in support of our hypothesis. As depicted in Figure 4, upon entering stage 2, the model's performance (red line) improves, reaching a peak AIME25 accuracy of 25.6%, a notable improvement over the Baseline GRPO model. Critically, this performance gain occurs even as the model's reliance on the parallel structure decreases (as shown by the declining parallel ratio in stage 2). This key observation suggests that the value of parallel thinking lies not only in the effectiveness of the parallel structure itself (which already outperforms the baseline), but more importantly, in the robust policy space it helps discover through exploration. The initial forced exploration acted as a scaffold, guiding the model to a more effective region in the policy space, from which it could then learn a final policy.

4 Conclusion

In this work, we presented **Parallel-R1**, the **first reinforcement learning framework** to teach large language models to perform parallel thinking from scratch on real-world mathematical reasoning tasks. We proposed a **progressive training curriculum**, enabled by a simple and scalable data pipeline, that successfully bootstraps this complex skill by separating the learning of format, behavior, and core reasoning into distinct stages. Our approach achieved consistent accuracy improvements on several challenging mathematical reasoning benchmarks compared to strong baselines.

Our analysis yielded several key insights into the learning dynamics. We discovered that the model learns a risk-averse strategy, shifting its use of parallel thinking from early-stage **computational exploration** to late-stage **multi-perspective verification**. Most significantly, we empirically identified and validated the potential of parallel thinking as a **mid-training scaffold**, showing that adding this temporary, forced-exploration phase can unlock higher final performance ceilings after RL training.

References

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Keyu Chen, Zhifeng Shen, Daohai Yu, Haoqian Wu, Wei Wen, Jianfeng He, Ruizhi Qiao, and Xing Sun. Aspd: Unlocking adaptive serial-parallel decoding by exploring intrinsic parallelism in llms. *arXiv preprint arXiv:2508.08895*, 2025a.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, et al. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025b.

Andy Clark. *Microcognition: Philosophy, cognitive science, and parallel distributed processing*, volume 6. MIT Press, 1989.

Runpeng Dai, Run Yang, Fan Zhou, and Hongtu Zhu. Breach in the shield: Unveiling the vulnerabilities of large language models. *arXiv preprint arXiv:2504.03714*, 2025.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv preprint arXiv:2508.15260*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

Chan-Jan Hsu, Davide Buffelli, Jamie McGowan, Feng-Ting Liao, Yi-Chang Chen, Sattar Vakili, and Da-shan Shiu. Group think: Multiple concurrent reasoning agents collaborating at token level granularity. *arXiv preprint arXiv:2505.11107*, 2025.

Chengsong Huang, Langlin Huang, and Jiaxin Huang. Divide, reweight, and conquer: A logit arithmetic approach for in-context learning. *arXiv preprint arXiv:2410.10074*, 2024.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Efficient test-time scaling via self-calibration. *arXiv preprint arXiv:2503.00031*, 2025a.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*, 2025b.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025c.

Ray Jackendoff. The parallel architecture and its place in cognitive science. *Syntax and Morphology Multidimensional. Eds. A. Nolda, O. Teuber. Berlin, New York: Mouton De Gruyter*, pages 17–44, 2011.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025a.

Tian Jin, Ellie Y Cheng, Zack Ankner, Nikunj Saunshi, Blake M Elias, Amir Yazdanbakhsh, Jonathan Ragan-Kelley, Suvinay Subramanian, and Michael Carbin. Learning to keep a promise: Scaling language model decoding parallelism with learned asynchronous decoding. *arXiv preprint arXiv:2502.11517*, 2025b.

Zongxia Li, Yapei Chang, Yuhang Zhou, Xiyang Wu, Zichao Liang, Yoo Yeon Sung, and Jordan Lee Boyd-Graber. Semantically-aware rewards for open-ended r1 training in free-form generation. *arXiv preprint arXiv:2506.15068*, 2025a.

Zongxia Li, Wenhao Yu, Chengsong Huang, Rui Liu, Zhenwen Liang, Fuxiao Liu, Jingxi Che, Dian Yu, Jordan Boyd-Graber, Haitao Mi, et al. Self-rewarding vision-language model via reasoning decomposition. *arXiv preprint arXiv:2508.19652*, 2025b.

Bo Liu, Leon Guertler, Simon Yu, Zichen Liu, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, et al. Spiral: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. *arXiv preprint arXiv:2506.24119*, 2025.

Thang Luong and Edward Lockhart. Advanced version of gemini with deep think officially achieves gold medal standard at the international mathematical olympiad. https://deepmind.google/discover/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/, 2025. Accessed: 2025-07-30.

Matthew Macfarlane, Minseon Kim, Nebojsa Jojic, Weijia Xu, Lucas Caccia, Xingdi Yuan, Wanru Zhao, Zhengyan Shi, and Alessandro Sordoni. Instilling parallel reasoning into language models. In *2nd AI for Math Workshop @ ICML 2025*, 2025. URL https://openreview.net/forum?id=a3o4b3hkwp.

Jiayi Pan, Xiuyu Li, Long Lian, Charlie Snell, Yifei Zhou, Adam Yala, Trevor Darrell, Kurt Keutzer, and Alane Suhr. Learning adaptive parallel reasoning with language models. *arXiv preprint arXiv:2504.15466*, 2025.

Gleb Rodionov, Roman Garipov, Alina Shutova, George Yakushev, Erik Schultheis, Vage Egiazarian, Anton Sinitsin, Denis Kuznedelev, and Dan Alistarh. Hogwild! inference: Parallel llm generation via concurrent attention. *arXiv preprint arXiv:2504.06261*, 2025.

Mahdi Sabbaghi, Paul Kassianik, George Pappas, Yaron Singer, Amin Karbasi, and Hamed Hassani. Adversarial reasoning at jailbreaking time. *arXiv preprint arXiv:2502.01633*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Yucheng Shi, Wenhao Yu, Zaitang Li, Yonglin Wang, Hongming Zhang, Ninghao Liu, Haitao Mi, and Dong Yu. Mobilegui-rl: Advancing mobile gui agent through reinforcement learning in online environment. *arXiv preprint arXiv:2507.05720*, 2025.

Sree Harsha Tanneru, Dan Ley, Chirag Agarwal, and Himabindu Lakkaraju. On the hardness of faithful chain-of-thought reasoning in large language models. *arXiv preprint arXiv:2406.10625*, 2024.

Haozhe Wang, Long Li, Chao Qu, Fengming Zhu, Weidi Xu, Wei Chu, and Fangzhen Lin. To code or not to code? adaptive tool integration for math language models via expectation-maximization. *arXiv preprint arXiv:2502.00691*, 2025a.

Haozhe Wang, Chao Qu, Zuming Huang, Wei Chu, Fangzhen Lin, and Wenhu Chen. Vl-rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning. *arXiv preprint arXiv:2504.08837*, 2025b.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025c.

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

Guangzhi Xiong, Qiao Jin, Xiao Wang, Yin Fang, Haolin Liu, Yifan Yang, Fangyuan Chen, Zhixing Song, Dengyu Wang, Minjia Zhang, et al. Rag-gym: Optimizing reasoning and search agents with process supervision. *arXiv preprint arXiv:2502.13957*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Xinyu Yang, Yuwei An, Hongyi Liu, Tianqi Chen, and Beidi Chen. Multiverse: Your language models secretly decide how to parallelize and merge generation. *arXiv preprint arXiv:2506.09991*, 2025b.

Xinyu Yang, Tianqi Chen, and Beidi Chen. Ape: Faster and longer context-augmented generation via adaptive parallel encoding. *arXiv preprint arXiv:2502.05431*, 2025c.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *arXiv preprint arXiv:2406.07394*, 2024.

Tong Zheng, Lichang Chen, Simeng Han, R Thomas McCoy, and Heng Huang. Learning to reason via mixture-of-thought for logical reasoning. *arXiv preprint arXiv:2505.15817*, 2025.

Yujun Zhou, Zhenwen Liang, Haolin Liu, Wenhao Yu, Kishan Panaganti, Linfeng Song, Dian Yu, Xiangliang Zhang, Haitao Mi, and Dong Yu. Evolving language models without labels: Majority drives selection, novelty promotes variation. *arXiv preprint arXiv:2509.15194*, 2025a.

Yujun Zhou, Jiayi Ye, Zipeng Ling, Yufei Han, Yue Huang, Haomin Zhuang, Zhenwen Liang, Kehan Guo, Taicheng Guo, Xiangqi Wang, et al. Dissecting logical reasoning in llms: A fine-grained evaluation and supervision study. *arXiv preprint arXiv:2506.04810*, 2025b.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

A Related Work

A.1 Parallel Thinking

Parallel thinking has emerged as an active area of research recently [Yao et al., 2023, Wang et al., 2022, Brown et al., 2024, Zhang et al., 2024, Huang et al., 2025a, Pan et al., 2025, Huang et al., 2024, Hsu et al., 2025, Rodionov et al., 2025, Yang et al., 2025b, Jin et al., 2025b]. Among them, a common brute-force strategy is to spawn multiple independent trajectories at the very beginning and join their outcomes only at the end [Brown et al., 2024, Wang et al., 2022], or to exchange thoughts at fixed intervals [Rodionov et al., 2025, Hsu et al., 2025]. Obviously, such schemes lack adaptivity as the points of branching and aggregating are dictated by a pre-defined schedule, not conditioned on the intermediate progress of the thinking process itself. To achieve finer-grained control, methods such as Monte Carlo Tree Search [Zhang et al., 2024] and Tree of Thoughts [Yao et al., 2023] offer more nuanced parallelism; however, they are still guided by hand-crafted heuristics based on external verifiers. More recent work [Pan et al., 2025, Yang et al., 2025b] strives for adaptivity through RL or SFT. However, these studies either (i) focus mainly on efficiency—losslessly converting a single long chain-of-thought into an adaptive parallel form via SFT, which limits the discovery of new reasoning patterns, or (ii) demonstrate RL only on toy tasks such as Countdown. In this work, we argue that learning parallel thinking via RL is a more generic and promising direction: it not only retains efficiency but also uncovers novel, highly adaptive reasoning behaviors, leading to improved performance beyond the "lossless transformation" paradigm of Yang et al. [2025b]. To this end, we proposed the first RL framework to stimulate adaptive parallel thinking for general mathematical tasks.

A.2 Improving Reasoning via RLVR

Reinforcement Learning with Verifiable Rewards (RLVR) optimizes language models via reinforcement learning using outcome-based, automatically checkable rewards, eliminating the need for trained reward models or step-level human annotations. Recent advances have demonstrated RLVR's effectiveness across diverse domains—including mathematical problem solving [Guo et al., 2025], coding [Wang et al., 2025a], multi-modal reasoning [Huang et al., 2025c, Wang et al., 2025b, Zheng et al., 2025, Li et al., 2025b], Search [Jin et al., 2025a, Xiong et al., 2025] text generation [Li et al., 2025a] and interactive GUI navigation [Shi et al., 2025]. In parallel, a growing body of work aims to make RLVR more efficient and stable, proposing new training paradigms such as self-play [Liu et al., 2025, Huang et al., 2025b] and test-time RL [Zuo et al., 2025, Zhou et al., 2025a], as well as more robust RL algorithms including DAPO [Yu et al., 2025], VAPO [Yue et al., 2025], and high entropy guided optimization [Wang et al., 2025c]. However, important challenges remain. Existing methods often leave unresolved issues of faithfulness [Tanneru et al., 2024, Chen et al., 2025b, Zhou et al., 2025b] and robustness [Sabbaghi et al., 2025, Dai et al., 2025]. Moreover, most approaches adopt a strictly sequential reasoning paradigm. This limitation is fundamental, as LLMs do not inherently possess parallel thinking capabilities, making it a significant and unresolved challenge to instill this skill using standard RLVR methods. To address this specific challenge, we introduce the first reinforcement learning framework that leverages a progressive curriculum to effectively instill the parallel thinking ability in LLMs, fundamentally enhancing their intrinsic reasoning capabilities.

B Prompts

Baseline Prompt

```
{Problem} Let's think step by step and output the final answer after "Final
    Answer: ".
```

Parallel Thinking Prompt

```
Solve the following problem step by step.
During the reasoning process, whenever you encounter a step that may benefit
    from multiple perspectives or independent reasoning, insert a <Parallel>
     block at that point.

Within each <Parallel> block:

Include at least two distinct and independent reasoning paths.
Each path must be enclosed within <Path> and </Path> tags.
Do not include any ordering information or cross-references between paths, as
     they are generated simultaneously and independently.
Close the block with </Parallel>.
Immediately after each </Parallel>, write a concise summary of insights or
    conclusions drawn from all paths, enclosed in <Summary> and </Summary>
    tags.

Repeat this process adaptively as needed throughout the reasoning.
Do not explicitly mention that you are triggering parallel thinking-just
    insert the <Parallel> block naturally within the reasoning chain.

End your response with a line starting with Final Answer: followed by the
    final result.

Problem: {Problem}
```

C Case Studies

Case Study: Find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b.

First, the number 17_b in base b is equal to $1 \times b + 7 = b + 7$ in base 10. Similarly, 97_b is equal to $9 \times b + 7 = 9b + 7$ in base 10. We need to find all integer bases $b > 9$ such that $b + 7$ divides $9b + 7$. At this point, consider two distinct strategies for finding b:

```
<Parallel>
<Path>
Factor theorem approach: Consider 9b + 7 as a polynomial in b
    and use the factor theorem.
Since b + 7 divides 9b + 7, by the factor theorem, 9(-7) + 7 =
    -63 + 7 = -56 must be divisible by b + 7.
Since b > 9, b + 7 > 16, and -56 can be factored as -56x1,
    -28x2, -14x4, or -7x8 (and their positive counterparts).
Only b + 7 = 14 gives b = 7, but b > 9, so no solution from
    this path.
However, we should also consider the positive factors for
    completeness.
The positive factorization 56x1, 28x2, 14x4, or 7x8 yields b +
    7 = 56 (giving b = 49),
b + 7 = 28 (giving b = 21), b + 7 = 14 (giving b = 7, invalid),
and b + 7 = 8 (giving b = 1, invalid).
Thus, the only valid solution from this path is b = 21 and b =
    49.
</Path>
<Path>
Using base arithmetic: In base b, 97\_b - 1x17\_b = 97\_b -
    17\_b = 80\_b,
which is 8xb + 0 = 8b in base 10.


Since 9b + 7 = 8b + (b + 7), b + 7 must divide 8b, implying
    that for some integer k,
9b + 7 = (b + 7)(8 + k).
Simplifying, 9b + 7 = 8b + 8k + kb + 7k, so b(1 - k) = 8k + 7.
Since b > 9, 1 - k must be positive or negative to keep b an
    integer.
If 1 - k = 1, then b = 8k + 7 with k < 1.
Possible integer k values are k <= 0, but k = 0 gives b = 7 <=
    9,
and k < 0 gives negative b, which is invalid.
Thus, this path leads to no valid solutions directly.
However, it reinforces that b + 7 must divide 8b.
</Path>
</Parallel>
<Summary>
These two approaches consider both the algebraic factorization
    method
and an alternative numerical or arithmetic approach to find
    possible values of b,
but only the factorization method yields valid bases greater
    than 9.
</Summary>
```

Next, consider the condition that $b + 7$ divides $9b + 7$. From the earlier, $9b + 7 = 9(b + 7) - 56$. For $b + 7$ to divide $9b + 7$, it must divide 56. Since $b + 7 \geq 17$ for $b > 9$, the only divisors of 56 exceeding 16 are 28 and 56, giving $b + 7 \in \{28, 56\}$ and thus $b \in \{21, 49\}$.
Finally, verify:
For $b = 21$: $17_{21} = 21 + 7 = 28$, $97_{21} = 9 \times 21 + 7 = 196$, $196 \div 28 = 7$.
For $b = 49$: $17_{49} = 49 + 7 = 56$, $97_{49} = 9 \times 49 + 7 = 448$, $448 \div 56 = 8$.

Thus, the sum is $21 + 49 = 70$. Final Answer: 70

Figure 5: A qualitative case study showcases that our model uses parallel exploration of two distinct algebraic methods for solving a number theory problem, both converging on the correct answer.

Case Study: Find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b.
First, I need to find all integer bases $b > 9$ for which 17_b is a divisor of 97_b. To do this, I'll start by converting these base-b numbers to base 10, as that's more familiar.
In base b, the number 17_b means $1 \times b + 7 = b + 7$ in base 10. Similarly, 97_b means $9 \times b + 7 = 9b + 7$ in base 10.
So, the condition that 17_b divides 97_b translates to $b + 7$ dividing $9b + 7$ in base 10.
Now, for $b + 7$ to divide $9b + 7$, the remainder when $9b + 7$ is divided by $b + 7$ must be zero. I can perform polynomial division or use the remainder theorem to find this.
Let me compute the remainder of $9b + 7$ divided by $b + 7$.
Divide $9b + 7$ by $b + 7$: - The quotient is 9, since $9 \times (b + 7) = 9b + 63$. - Subtract this from $9b + 7$: $(9b + 7) - (9b + 63) = 9b + 7 - 9b - 63 = -56$.
So, the remainder is -56. For $b + 7$ to divide $9b + 7$, it must divide -56, or equivalently, $b + 7$ must be a positive divisor of 56.
The positive divisors of 56 are: 1, 2, 4, 7, 8, 14, 28, 56.
Since $b > 9$, we have $b + 7 > 16$. So, $b + 7$ must be a divisor greater than 16. - $b + 7 = 28 \Rightarrow b = 21$ (valid) - $b + 7 = 56 \Rightarrow b = 49$ (valid)
So, the valid bases are $b = 21$ and $b = 49$. Now, I need to find the sum: $21 + 49 = 70$.
But before finalizing, I should confirm that these are the only valid bases. From the divisor check, only $b + 7 = 28$ and $b + 7 = 56$ give $b > 9$, so no others.

```
<Parallel>
<Path>One might consider that $ b + 7 $ not only divides 56 but
    also that it must be such that $ 17_b $ and $ 97_b $ are
    valid base-$b$ numbers, which requires $ b > 7 $ (since
    digits 9 and 7 are used). However, since $ b > 9 $ is given,
    this is already satisfied, and the divisor condition is the
    key.</Path>
<Path>Use the fact that for $ b + 7 $ to divide $ 9b + 7 $, the
    remainder must be zero, so we can solve $ 9b + 7 = k(b + 7)
    $ for integer $ k $, but it's simpler to compute the
    remainder directly as above.</Path>


</Parallel>
<Summary>
The parallel block explored alternative strategies for solving
    $ b + 7 $ divides $ 9b + 7 $, including finding divisors,
    considering base validity, and solving equations, but
    ultimately confirmed that only $ b = 21 $ and $ b = 49 $
    satisfy the condition for $ b > 9 $.
</Summary>
```

Now, sum the valid bases: $21 + 49 = 70$. Final Answer: 70

Figure 6: A qualitative case study demonstrates late-stage verification through parallel thinking.

Table 4: Ablation Study on Training Approach: Comparison of different training configurations.

Training Configuration	AIME25	AIME24	AMC23	MATH	Avg.
Effect of Training Stages					
Parallel-R1-Seen	**19.2**	**19.4**	**70.5**	**86.7**	**48.9**
- w/o RL on GSM8K	17.9	19.0	65.0	84.5	46.6
Parallel-R1-Unseen (S1)	17.7	18.3	69.7	82.6	47.1
+ with RL on GSM8K	14.4	12.9	52.3	74.4	38.5

D More Ablations

D.1 Ablation on Training Approach

We further investigate the role of two-stage RL in our training pipeline. One natural question is whether learning on GSM8K, which is a relatively simple math dataset, truly benefits from the RL, given that the structural parallel reasoning format (e.g., the correct use of `<Parallel>`, `<Path>`, and `<Summary>` tokens) can be directly acquired through SFT [Yang et al., 2025b].

Algorithm 1 Parallel Thinking Format Check

Input: tokens – list of tokens from the parallel-thinking trace;
tag_pairs – set of valid (opening, closing) tag pairs, e.g. {(<Path>...</Path>), ...}
Output: format_valid – boolean indicating whether the trace is well-formed

$S \leftarrow \emptyset$
$format\_valid \leftarrow$ true
for all t **in** tokens **do**
  if t **is an opening tag then**
    push t onto S
  else if t **is a closing tag then**
    if S is empty **then**
      $format\_valid \leftarrow$ false
      break
    end if
    $top\_tag \leftarrow \text{Top}(S)$
    if $(top\_tag,\ t) \in$ tag_pairs **then**
      pop S
    else
      $format\_valid \leftarrow$ false
      break
    end if
  end if
end for
if $format\_valid$ **and** $S \neq \emptyset$ **then**
  $format\_valid \leftarrow$ false
end if
return $format\_valid$

Table 5: Ablation Study on Parallel Thinking Prompt.

Training Configuration	AIME25	AIME24	AMC23	MATH	Avg.
Effect of Parallel Thinking Prompt					
Parallel-R1-Seen	19.2	**19.4**	**70.5**	**86.7**	**48.9**
- w/o Parallel Thinking Prompt	**20.4**	16.5	66.7	84.8	47.1

Table 4 presents the ablation results. For the Parallel-Seen variant, keeping the Cold Start SFT but removing stage one RL training on GSM8K leads to a consistent performance drop (–2.3% on average). This indicates that learning format through SFT alone is insufficient. Without stage one RL, the model enters stage two training on general math without having acquired the ability to trigger or use parallel thinking adaptively. As a result, RL training must simultaneously learn both adaptive parallel thinking behavior and mathematical reasoning ability, which is harder to optimize.

Interestingly, the Structure variant exhibits the opposite trend: adding stage one RL on GSM8K severely hurts performance (–8.6% on average). We hypothesize that this is because the structured attention mask learned on easy math tasks (GSM8K) does not transfer well to the distribution shift of harder math problems, causing overfitting to superficial patterns, which is consistent with findings in [Yang et al., 2025c]. This contrast highlights a key insight: while stage one RL is crucial for the causal variant to bootstrap adaptive parallel thinking, structural variants require a different training recipe and reward schedule to generalize effectively.

D.2 Ablation on Parallel Thinking Prompt

We also conduct an ablation study on the effect of our parallel thinking prompt. As shown in Table 5, removing the prompt leads to a performance degradation of up to 1.8% on average. It indicates that providing more detailed instructions during training helps the model better understand the reasoning process, rather than merely memorizing the output patterns.