Parason: Revealing Subtask- and Trial Parallelism in LLM Reasoning

Zhengyang Zhang[1] **Zijian Zhang**[1 4] **Jiaxuan Gao**[1]
Shusheng Xu[2] **Yi Wu**[1] **Song Han**[3 4] **Ligeng Zhu**[4 *]

[1] Tsinghua University [2]Independent Researcher [3]Massachusetts Institute of Technology [4] NVIDIA

Abstract

Scaling test-time reasoning has substantially improved the problem-solving ability of large language models (LLMs), but standard autoregressive decoding still executes long reasoning traces sequentially, creating severe latency for difficult tasks (up to days and weeks). Parallel reasoning offers a natural remedy. However, prior systems primarily focus on *Subtask Parallelism*, where the model learns to decompose a high-level task into smaller chunks that can be solved independently. This approach overlooks another pervasive form of parallelism: *Trial Parallelism*, where multiple speculative attempts explore, verify, and aggregate competing hypotheses in parallel. In this paper, we introduce **Parason**, which reveals and learns both forms of parallelism in LLM reasoning. Our analysis identifies Trial Parallelism as the majority of parallelizable reasoning computation (65.5% in DeepSeek-V4's reasoning steps in HLE), and it becomes increasingly dominant on hard problems. Guided by this taxonomy, Parason converts sequential reasoning traces into structured parallel trajectories with a context-free grammar, then trains models with Parallelism-Aware Group Relative Policy Optimization (PA-GRPO), whose reward jointly balances accuracy, latency, and the two parallelism ratios. At inference time, Parason executes the learned parallel structure through tool calls, translating theoretical savings to real-world wall-clock acceleration. Experiments on mathematical reasoning benchmarks including AIME24 and AIME25 show that Parason achieves **an average acceleration about 1.7**$\times$ while maintaining competitive accuracy.

Website | **Code** | **Dataset**

1 Introduction

Recent advances in large language model (LLM) reasoning have made a simple recipe increasingly effective: spend more test-time compute, and obtain better answers. Chain-of-Thought prompting [Wei et al., 2022] and reasoning-oriented systems such as OpenAI o1 [OpenAI et al., 2024] and DeepSeek-R1 [DeepSeek-AI et al., 2025] show that long, deliberate reasoning traces can substantially improve performance on mathematical and logical tasks. Yet this recipe has an immediate systems bottleneck. Standard autoregressive decoding serializes every token and every intermediate thought, so stronger reasoning often translates directly into longer waiting time. On difficult problems, reasoning traces can grow to hundreds of thousands or even millions of tokens [Hubert et al., 2025, Luong et al., 2025], making the usual "think longer" strategy impractical for interactive agents, coding assistants, and other latency-sensitive applications. For example, Google's AlphaProof [DeepMind, 2024] requires up to *three days* to solve challenging problems from the International Mathematical Olympiad (IMO) 2024, highlighting the extreme latency inherent in current sequential reasoning systems.

*ligengz@nvidia.com

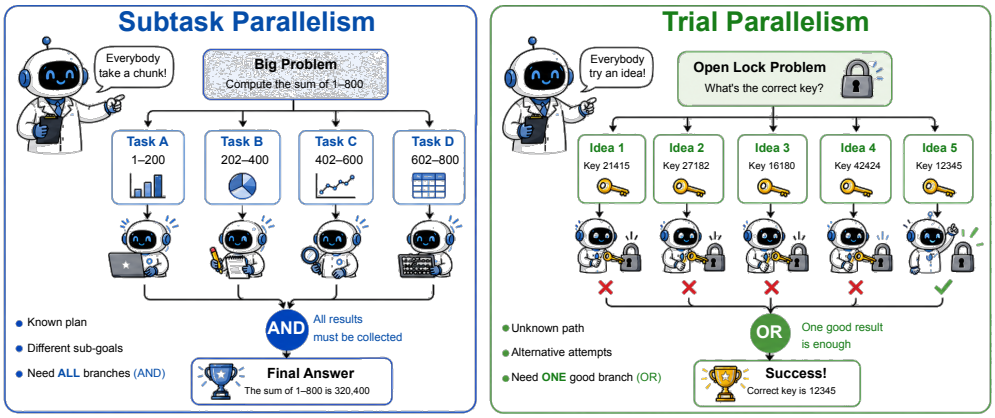

Figure 1: **Overview of two types of parallelism.** The figure contrasts Subtask Parallelism, where all decomposed branches are needed for the final answer, with Trial Parallelism, where multiple uncertain paths are explored as alternative (OR) branches. Despite this OR relation, the outputs of all Trial branches are concatenated into the subsequent context, making every branch's content available for final synthesis. While previous work mainly focuses on Subtask Parallelism, our analysis shows that Trial Parallelism accounts for the majority of parallelizable reasoning on HLE for every model we study, and exceeds 58% for DeepSeek-R1 and DeepSeek-V4 across both datasets.

A natural response is to make reasoning parallel: let multiple reasoning branches proceed concurrently instead of decoding every thought in sequence. Existing methods explore this direction through independent sampling, such as self-consistency and Best-of-N [Wang et al., 2022, Hu et al., 2026], or adaptive branching frameworks [Yang et al., 2025b, Jin et al., 2025, Zheng et al., 2025, Lian et al., 2025, Wang et al., 2025]. However, methods such as Multiverse [Yang et al., 2025b], ThreadWeaver [Lian et al., 2025], and APR [Pan et al., 2025] mainly exploit *subtask-style* parallelism: splitting a high-level task into smaller chunks and merging their results. Trial-style exploration, which tries uncertain paths and incorporates diverse trajectories, is less discussed.

In this work, we argue that the key missing piece is a semantic taxonomy of parallel reasoning. As illustrated in Figure 1, we identify two complementary forms of parallelism. **Subtask Parallelism** applies when a problem can be decomposed into independent steps: each branch solves a distinct sub-goal, and the final answer combines all branch results. This is the form emphasized by most prior adaptive parallel systems. **Trial Parallelism** applies when the model is uncertain about which path will work: multiple branches test competing hypotheses, and the final trajectory keeps the useful results. These modes have different execution semantics. Subtask branches are usually all necessary, while trial branches are speculative and most useful when the reasoning path is uncertain. Our empirical analysis shows that this distinction is practical, not merely conceptual. Trial Parallelism accounts for the majority of analyzed parallelizable steps on Humanity's Last Exam for every model we study, including 73.8% and 65.5% for DeepSeek-R1 and DeepSeek-V4, respectively. On OpenMath, it remains the majority for most models and reaches 68.5% and 58.1% for the same two open-source models. This suggests that hard reasoning is not only decomposition, but also trying and refining uncertain solution paths. Systems that focus solely on subtask decomposition therefore miss much of the computation in hard cases.

Guided by this observation, we introduce **Parason**, a training-and-inference codesign framework for revealing and exploiting both Subtask Parallelism and Trial Parallelism. Parason converts sequential reasoning traces into structured parallel trajectories without changing their final semantics, and uses a context-free grammar to mark parallel regions, individual branches, and summaries in an engine-parseable format. We then train the model with Parallelism-Aware Group Relative Policy Optimization (PA-GRPO), which augments outcome-based reinforcement learning with rewards for lower latency along the longest token path and balanced use of the two parallelism types. At inference time, the generated structure is executed through tool calls, allowing Parason to dispatch independent workers with subtasks and competing trial branches. Across mathematical reasoning benchmarks including AIME24 and AIME25, Parason preserves competitive accuracy while reducing effective reasoning latency. Our contributions are summarized as follows:

Sequential Reasoning is Naturally Parallelizable – A Case of 24 Puzzle Solving

Prompt: Calculate 24 using 6 9 9 10

<think> We are given the numbers 6, 9, 9, 10 and we need to make 24 using basic arithmetic operations (+, -, *, /) and possibly parentheses. We need to find an expression that equals 24. Let's try different combinations: We have two 9's, one 6, and one 10. Goal: 24. Some common approaches: 24 = 4*6, 24=30-6, 24=2.4*10, etc.

Since we have a 6, maybe we can get 4 from 9,9,10 to multiply by 6? 4*6=24. How to get 4 from 9,9,10? 10 - 9 = 1, then 9 - 1 = 8, not 4. Or (9+9)/10? =18/10=1.8, not 4. Maybe 24 = 30 - 6. So if we can make 30 from 9,9,10 and then subtract 6? But we have to use all numbers. So maybe (9+9+10) = 28, then 28 - 4? But we have 6. Alternatively, (9*10)/? Not sure. Or 24 = (10-6)*6, but we only have one 6. Maybe using fractions: 24 = 9/(something) etc. Let's try other combinations systematically: We can think of possible expressions: One idea: 10 * (9 - 6) = 10*3=30, then 30 - 9 = 21, not 24. 10 * (9 - 6) - 9 = 30-9=21. 9 * (10 - 6) = 9*4=36, then 36 - 9 = 27, not 24. (9+9)* (10-6)? =18*4=72 too big. (9+9-10)*6 = (8)*6=48, too big.

Maybe divide: (9*10)/ (something) = 90/ something. 90/4 = 22.5, not 24. Or (9*10*9)/ something. 810/33.75 = 24? Not likely. Maybe using subtraction and division: 24 = (10*9)/(something) - something. Let's try: 9*10 = 90, 90/6=15, then 15+9=24! That works: 9*10/6+9 = 90/6+9=15+9=24. But we have to use two 9's and one 10 and one 6. In that expression we used one 9, one 10, one 6, and then added the other 9. So it's 9*10/6+9. But check order of operations: multiplication and division first: 9*10=90, 90/6=15, 15+9=24. Yes.

So the expression is: $9 \times 10 \div 6 + 9 = 24$. we found a solution. But let's check if there are other ways: maybe $(9 - 6) \times (10 - \sqrt{9})$ but we cannot use square root unless specified. So basic operations only. Another: (10 - 9/9)*? (10-1)=9, 9*? Not 24. (9+9+10-6)? =22, not 24.(9*9-10)/6? (81-10)/6=71/6≈11.83, no.

So the one we found seems simplest. Thus, the answer is: $9 \times 10 \div 6 + 9 = 24$. </think>

One solution is: $9 \times 10 \div 6 + 9 = 24$

Figure 2: **Visualization of Trial and Subtask Parallelism.** The 24 Puzzle trace generated by DeepSeek-R1 [DeepSeek-AI et al., 2025] highlights Subtask Parallelism in blue and Trial Parallelism in green. Subtask branches implement divide-and-conquer execution, while Trial branches explore and verify competing hypotheses; *Trial steps occupy the majority* of reasoning tokens.

- **A taxonomy of parallel reasoning.** We distinguish **Subtask Parallelism** and **Trial Parallelism**, identifying Trial Parallelism as the missing component. Based on this taxonomy, we introduce Parason, which converts sequential reasoning data into grammar-constrained parallel trajectories that can be parsed and executed by inference engines.
- **Parallelism-aware reinforcement learning.** We propose PA-GRPO, a multi-objective RL objective that jointly optimizes answer accuracy, latency along the longest token path, and the model's use of the two parallelism modes.
- **Empirical speedups with competitive accuracy.** Experiments across challenging math benchmarks demonstrate that Parason improves the latency–accuracy Pareto frontier, achieving an average acceleration about $1.7\times$ while maintaining competitive accuracy. Under latency-constrained settings, Parason matches the performance of an 8k-token latency budget using only 25% of the budget.
- **Seamless inference engine support.** We define a CFG that marks parallel regions as tool calls and integrate Parason into SGLang [Zheng et al., 2023]. This makes parallel reasoning directly executable in a real inference engine, while prior work either reports only theoretical speedups that are hard to translate into actual latency reduction, or introduces complicated designs that require deep modifications to modern inference engines.

2 Related Work

Test-time scaling for LLM reasoning. Scaling inference-time computation has become a central recipe for improving LLM reasoning. CoT prompting [Wei et al., 2022] elicits intermediate derivations, while recent reasoning models such as OpenAI o1 [OpenAI et al., 2024] and DeepSeek-R1 [DeepSeek-AI et al., 2025] further improve performance by producing longer and more deliberate reasoning traces. This sequential scaling is powerful but expensive: autoregressive decoding generates every token one after another, so longer reasoning directly increases latency. Parallel reasoning is an attractive direction for preserving test-time compute while shortening the longest token path.

Independent parallel sampling and majority voting. A simple form of parallel reasoning is to sample multiple complete solutions and aggregate their answers. Self-consistency [Wang et al.,

Model	Dataset: OpenMath		Dataset: HLE	
	Subtask (%)	Trial (%)	Subtask (%)	Trial (%)
Open Source Models				
DeepSeek-R1 [DeepSeek-AI et al., 2025]	31.5	$68.5_{+37.00}$	26.2	$73.8_{+47.60}$
Qwen3-30B-A3B [Yang et al., 2025a]	43.6	$56.3_{+12.70}$	38.8	$61.2_{+22.40}$
MiniMax 2.7 [MiniMax AI, 2025]	47.7	$52.3_{+4.60}$	38.2	$61.8_{+23.60}$
Kimi-2.6 [Moonshot AI, 2025]	43.0	$57.0_{+14.00}$	42.1	$57.9_{+15.80}$
DeepSeek-V4 [DeepSeek-AI, 2026]	41.9	$58.1_{+16.20}$	34.5	$65.5_{+31.00}$
Commercial Models				
Gemini-2.5-Pro [Google, 2025]	62.9	$37.1_{-25.80}$	49.4	$50.6_{+1.20}$
Claude-Opus-4.5 [Anthropic, 2025b]	44.9	$55.1_{+10.20}$	29.9	$70.1_{+40.20}$
Gemini-3-Pro [Google DeepMind, 2025]	57.4	$42.6_{-14.80}$	46.9	$53.1_{+6.20}$
GPT-5.5 [OpenAI, 2026]	34.4	$65.6_{+31.20}$	23.9	$76.1_{+52.20}$

Table 1: **Trial Parallelism dominates on harder reasoning datasets.** We report Subtask and Trial ratios in reasoning traces from OpenMathReasoning [Moshkov et al., 2025] and Humanity's Last Exam (HLE) [Phan et al., 2025]. Ratios are computed on the first 100 problems of each dataset, and annotation details are provided in Appendix A. For closed-source models, the original thinking trace is hidden, thus we counted summary statistics instead.

2022], verifier-guided Best-of-N sampling [Cobbe et al., 2021], and confidence-based variants such as DeepConf [Fu et al., 2025] improve robustness by exploring several candidate trajectories. These methods implicitly use trial-style computation, since different samples may try different solution paths. However, these samples are independent full traces. They do not share intermediate work, expose branch structure inside a trace, or tell an inference engine where parallel execution should begin and end. As a result, they improve accuracy at the cost of redundant computation with little control over latency.

Structured and adaptive parallel reasoning. Another line of work introduces explicit structures for parallel reasoning. Tree-of-Thoughts [Yao et al., 2023], Graph-of-Thoughts [Besta et al., 2023], Skeleton-of-Thought [Ning et al., 2023], and agentic decomposition methods split reasoning into trees, graphs, outlines, or sub-agents. More recent adaptive systems, including PASTA [Jin et al., 2025], Multiverse [Yang et al., 2025b], Parallel-R1 [Zheng et al., 2025], APR [Pan et al., 2025], ThreadWeaver [Lian et al., 2025], and PaCoRe [Hu et al., 2026], train or prompt models to create parallel branches and merge their outputs. PaCoRe further scales test-time compute with multiple rounds of coordinated parallel exploration and message passing. Figure 2 visualizes the same broader opportunity: *LLM reasoning can be parallelized*.

Yet their main mechanism is still *subtask-centric*: the model decomposes a high-level task into smaller chunks, runs the chunks in parallel, and merges the results. This is effective when the problem admits a clean divide-and-conquer structure, but it misses a common behavior in difficult reasoning: trying uncertain paths, rejecting failed attempts, and keeping the path that works. Our analysis shows that this missing *Trial Parallelism is not rare*; it accounts for most parallelizable reasoning steps in hard cases. Parason therefore separates **Subtask Parallelism** from **Trial Parallelism** and trains the model to exploit both forms explicitly.

3 Methodology

Parason turns long sequential reasoning traces into executable parallel programs, which is critical for long-horizon tasks such as Google's AlphaProof [DeepMind, 2024], OpenAI's IMO reasoning efforts [Luong et al., 2025], and recent agentic coding workflows such as Claude Code [Anthropic, 2025a] and OpenAI Codex [OpenAI, 2025]. The framework has three parts. First, we define two semantic forms of parallel reasoning: Subtask Parallelism decomposes a problem into independent required parts, while Trial Parallelism explores competing uncertain attempts and incorporates the diverse exploration into the final reasoning path. Second, we define a context-free grammar (CFG) that gives Parason a formal syntax, helps data generation, processing, and rollouts avoid syntax errors, and makes later system integration simple. Third, we propose Parallelism-Aware GRPO (PA-GRPO),

Parallel Reasoning in Parason – A Case of 24 Puzzle Solving

Prompt: Calculate 24 using 6, 9, 9, 10.

```
<think> We are given the numbers 6, 9, 9, 10 and need to make 24 using basic arithmetic operations (+, -, *, /).
Goal: 24.

<Parallel>
 <Outlines>
  <Subtask> 1. Try to use 6 as a multiplier and derive 4 from 9, 9, 10. </Subtask>
  <Subtask> 2. Try division with 9*10 and use the remaining 6 and 9. </Subtask>
  <Subtask> 3. Verify whether there are simpler alternatives. </Subtask>

 <Thread> 1.
  <Parallel>
   <Outlines>
    <Trial> 1. Try use 9, 9, 10 to
    make 4. 10-9=1, then 9-1=8, not 4.
    This route does not find 4.
    </Trial>
    <Trial> 2. Try 10*(9-6)-9. This
    gives 21, not 24. This route does
    not find 24. </Trial>
    <Trial> 3. Try 9*(10-6)-9. This
    gives 27, not 24. </Trial>
    <Trial> 4. Try (9+9)*(10-6). This
    gives 72, not 24. </Trial>
   </Outlines>
  </Parallel>
 </Thread>

 <Thread> 2.
  <Parallel>
   <Outlines>
    <Trial> 1. Try 9*10/6+9. 90/6=15,
    15+9=24. This works. </Trial>
   </Outlines>
  </Parallel>
 </Thread>

 <Thread> 3.
  <Parallel>
   <Outlines>
    <Trial> 1. Try (9-6)*(10-sqrt(9)).
    Square root is not allowed.
    </Trial>
    <Trial> 2. Try 9+9+10-6. Result
    is 22, not 24. </Trial>
    <Trial> 3. Try (9*9-10)/6. Result
    is 11.83, not 24. </Trial>
   </Outlines>
  </Parallel>
 </Thread>
</Outlines>

 </Outlines>
</Parallel>
Thus, the answer is 9 × 10 ÷ 6 + 9 = 24. </think>
```

Figure 3: **Solving the 24 Puzzle in Parason format.** This example is adapted from a real 24 Puzzle reasoning trace. The blue region gives the Subtask plan, splitting the problem into three independent subtasks. The green regions show Trial Parallelism, where each subtask explores candidate routes in parallel. The correct solution appears in the first Trial branch of the second subtask.

a training algorithm that teaches the model to use parallelism effectively while preserving final-answer correctness.

3.1 Parallelism in Reasoning

A reasoning trace is not uniformly sequential, as shown by the visualization of Trial and Subtask Parallelism in Figure 2. Some steps derive independent facts that will all be used later; other steps try uncertain ideas, explore multiple paths, and aggregate their findings. Following the overview and examples in Figures 1, 2, and 3, Parason treats these behaviors as two semantic forms of parallelism with different merge rules.

Subtask Parallelism is an *AND-branch* form of parallelism. A problem is split into independent sub-goals $g_1, \ldots, g_k$, each branch computes a necessary result r_i, and the main trajectory aggregates all results $\{r_i\}_{i=1}^{k}$ to continue the derivation. These branches are parallel because they do not depend on one another, but all of them are mandatory: dropping one branch would remove the information required for generating the final answer. For example, a geometry solution may compute two distances independently before combining them in a final formula. This is the dominant form assumed by prior adaptive parallel systems [Yang et al., 2025b, Lian et al., 2025, Pan et al., 2025, Hu et al., 2026].

Trial Parallelism is an *OR-branch* form of parallelism. Given an uncertain state, the model launches competing attempts $a_1, \ldots, a_k$ that test different hypotheses, heuristics, or solution routes. The merge step concatenates the branches to incorporate the exploratory trajectories into the reasoning history, rather than selecting a single branch. Thus, Trial Parallelism is parallel search inside a single reasoning trace, rather than decomposition into mandatory parts. This mirrors how strong reasoning models often behave on hard problems: they try a path, detect a dead end, and revise. Our empirical analysis in Table 1 shows that Trial Parallelism accounts for more than 50% of parallelizable steps for every model on HLE [Phan et al., 2025] and for most models on OpenMath, with higher Trial shares on HLE across all listed models.

Context-Free Grammar (CFG) for Parason

Let $G = (V, \Sigma, R, S)$ be the context-free grammar for structured parallel reasoning.

Terminology

- **Terminals (Σ):** {<think>, <Parallel>, <Outlines>, <Subtask>, <Trial>, <Thread>, their matching closing tags, Text}
- **Non-terminals (V):**
  - S: start of reasoning
  - B: reasoning body
  - P: parallel region
  - O: outlines block
  - G: list of outline entries
  - E: one Subtask or Trial entry
  - H: list of thread bodies
  - W: one Thread body
  - T: free-form reasoning text
- **Start symbol:** S
- **Rules:** R defines valid tag nesting and branch structure.

Production Rules R

S → <think> B </think>

B → T B | P B | ϵ

P → <Parallel> O H </Parallel>

O → <Outlines> G </Outlines>

G → E G | E

E → <Subtask> T </Subtask>
  | <Trial> T </Trial>

H → W H | W

W → <Thread> B </Thread>

T → nonempty free-form text excluding reserved tags

Figure 4: **Context-Free Grammar for Parason.** Each parallel region contains an Outlines block with one or more correctly matched Subtask or Trial entries, followed by one or more Thread bodies. As in Figure 3, each Thread may contain free-form reasoning or a nested parallel region.

3.2 Parallel Tracjectory Format

To make these two modes executable, Parason represents reasoning traces with a grammar-constrained format. We extend ordinary <think> traces with parallel tags: <Parallel> marks a parallel region, <Outlines> describes the purpose of the region, <Subtask> and <Trial> mark branch type, <Thread> stores branch outcome. The full CFG is shown in Figure 4. While real-world reasoning often exhibits mixed dependencies (e.g., a subtask internally spawning trial branches), our training data keeps a strict separation for simplicity. However, we observe that the model can generalize to these complex compositions during evaluation.

The format is designed around two requirements. First, it is *semantic*: the tags indicate whether a branch is a necessary subtask or a speculative trial branch, not merely that it can run in parallel. This matters because prior parallel-reasoning methods often mix the two forms under a single branching interface, making it hard to decide whether branches should be all merged or aggregated as exploration history. Second, it is *engine-parseable*: each branch has explicit start and stop tags, so an inference runtime can dispatch workers without modifying the model architecture. Compared with free-form summaries or prompt-only, the CFG gives the training and inference infra a shared contract.

3.3 Parallelism-Aware Reinforcement Learning

Supervised conversion teaches the model to imitate parallel traces, but it does not directly optimize latency or decide *when* and *what* parallelism is worth using. We therefore RL fine-tune with **Parallelism-Aware Group Relative Policy Optimization (PA-GRPO)**. The reward keeps correctness as the primary objective, while shaping the model toward useful parallel branches. For a sampled trajectory i, let T_i denote its token-level latency, and let R^i_{subtask} and R^i_{trial} be the ratios of tokens placed in Subtask and Trial branches. We define:

Definition 3.1. Parallel RL Reward

$$R_i = -1 + 2 \times \mathbb{1}(\text{correct}_i) \times \left(1 - \alpha f\left(\frac{T_i - \mu_T}{\sigma_T}\right) + \beta_{\text{subtask}} f\left(\frac{R^i_{\text{subtask}} - \mu_s}{\sigma_s}\right) + \beta_{\text{trial}} f\left(\frac{R^i_{\text{trial}} - \mu_r}{\sigma_r}\right) + \min(\rho \cdot \eta(s),\ \rho_{\text{clip}})\right).$$

The symbols in Eq. 3.1 are defined as follows:

- R_i: final reward for sampled trajectory i.
- $\mathbb{1}(\text{correct}_i)$: correctness indicator; incorrect answers receive the base negative reward.
- T_i: token-level latency of trajectory i, measured by its critical-path token count.
- α: coefficient controlling the normalized latency penalty.
- R^i_{subtask}: Number of tokens placed in Subtask Parallelism branches/Total generated tokens
- $\eta(s)$: $\eta(s) = 1 - \frac{L_{\text{latency}}}{L_{\text{total}}}$, where L_{latency} and L_{total} denote the number of latency tokens and the total number of tokens, respectively.
- ρ, ρ_{clip}: Parameters for controlling acceleration reward.
- R^i_{trial}: Number of tokens placed in Trial Parallelism branches/Total generated tokens
- β_{subtask} and β_{trial}: incentives for Subtask and Trial Parallelism.
- f: shaping function applied to normalized reward signals; here we use the linear function $y = x$.
- μ_T, μ_s, μ_r and $\sigma_T, \sigma_s, \sigma_r$: means and standard deviations used to normalize latency, Subtask ratio, and Trial ratio, respectively.

The first term makes correctness the primary signal by assigning a negative base reward to incorrect answers and a positive base reward to correct answers. The latency term, weighted by α, penalizes trajectories with high normalized critical-path latency. The Subtask and Trial incentives, weighted by β_{subtask} and β_{trial}, encourage useful parallel structure, while the clipped acceleration reward directly favors reductions in the critical path relative to total generation length. It is important to note that β_{subtask} and β_{trial} target fundamentally different optimization goals: Trial Parallelism improves accuracy by trading total tokens for search width, while Subtask Parallelism reduces latency by compressing the longest token path. In practice, this reward lets us separately tune these two dimensions, as detailed in Section 4.2.

3.4 Integration into Modern Inference Engine

Parason integrates with modern inference engines by treating parallel branches as tool calls rather than requiring deep runtime changes. Following the CFG in Figure 4, the runtime decodes normally until a `<Parallel>` region appears, parses the `<Outlines>`, dispatches each `<Subtask>` or `<Trial>` as an independent worker, and returns the resulting `<Thread>` outputs to the main trajectory; Figure 3 shows this format on the 24 Puzzle. XGrammar [Dong et al., 2024] enforces valid tags, and we implement this tool-call execution in SGLang [Zheng et al., 2023].

4 Experiments

We provide details on training data curation, training framework, and evaluation below, and will open-source our implementation to facilitate reproducibility.

Training Data Curation. We build parallel training data from the 964 annotated Qwen3-8B traces released by ThreadWeaver [Lian et al., 2025]. Gemini-3-Flash labels each parallel stage as Subtask or Trial; for Trial stages, Qwen3-8B samples extra branches and Gemini-3-Flash generates their `<Trial>` goals. For reinforcement learning (RL), we use Polaris-53k [An et al., 2025], a collection of about 53,000 complex reasoning problems.

Training Framework. We use a two-stage post-training pipeline. First, we fine-tune the model with TRL [von Werra et al., 2020] on the curated parallel trajectories. Second, we train with VeRL [Sheng

Method	AIME24	AIME25	Math500	AMC	Avg	Avg. Latency #Tokens
Parallel-R1 (4B)	19.4	19.2	-	-	-	-
ShorterBetter (7B)	53.3	-	-	-	-	-
ThinkPrune (32B)	72.5	-	93.8	**95.9**	-	-
DYNASOR-COT (32B)	<u>78.0</u>	-	93.0	94.0	-	-
Dynamic Early Exit (32B)	70.0	-	**94.8**	95.0	-	-
AdaptThink (7B)	55.6	-	92.0	-	-	-
Multiverse (32B)	53.8	45.8	91.8	-	-	-
ThreadWeaver (8B)	**79.9**	60.5	92.3	91.4	81.0	14.8k
Parason-8B (w/vanilla SFT)	73.5	67.9	93.4	93.9	82.2	13.2k
+ PA-GRPO ($\beta_{\text{subtask}} = 0.025$)	75.1	69.7	93.8	96.6	83.8	13.8k
+ PA-GRPO ($\beta_{\text{subtask}} = 0.050$)	76.3	<u>70.0</u>	**94.6**	95.0	84.0	15.6k
+ PA-GRPO ($\beta_{\text{subtask}} = 0.100$)	<u>77.3</u>	68.5	**94.6**	96.3	84.2	14.1k
+ PA-GRPO ($\beta_{\text{trial}} = 0.025$)	75.5	69.9	<u>94.4</u>	96.3	84.0	14.2k
+ PA-GRPO ($\beta_{\text{trial}} = 0.050$)	76.5	**70.6**	94.0	96.9	<u>84.5</u>	14.5k
+ PA-GRPO ($\beta_{\text{trial}} = 0.100$)	**78.2**	68.9	94.2	**97.5**	**84.7**	14.5k
Additional runs with $\alpha = 0.1$						
+ PA-GRPO ($\beta_{\text{subtask}} = 0.025$)	75.7	67.6	93.6	95.9	83.2	<u>12.2k</u>
+ PA-GRPO ($\beta_{\text{subtask}} = 0.050$)	74.2	66.1	93.8	95.9	82.5	<u>12.2k</u>
+ PA-GRPO ($\beta_{\text{subtask}} = 0.100$)	75.5	65.9	92.6	**97.5**	82.9	13.2k
+ PA-GRPO ($\beta_{\text{trial}} = 0.025$)	76.9	66.6	**94.6**	<u>97.2</u>	83.8	**12.1k**
+ PA-GRPO ($\beta_{\text{trial}} = 0.050$)	<u>77.3</u>	67.3	93.4	96.6	83.7	12.5k
+ PA-GRPO ($\beta_{\text{trial}} = 0.100$)	**78.2**	69.2	93.4	95.6	84.1	13.4k

Table 2: **Trial rewards improve accuracy, while Subtask rewards reduce latency.** We compare Parason with prior systems and ablate β_{subtask} and β_{trial}. Avg and Token Latency average AIME24, AIME25, Math500, and AMC for each PA-GRPO row. Bold and underlined values mark the highest and second-highest PA-GRPO results, except Token Latency, where lower is better.

et al., 2024] using Parallelism-Aware GRPO (PA-GRPO). Unless otherwise noted, RL uses learning rate $1e{-}6$, batch size 128, 8 rollouts, lower clip ratio 0.2, and upper clip ratio 0.28; SFT uses learning rate $1e{-}5$, batch size 16, 8 epochs, and a cosine learning-rate scheduler.

Evaluation. We evaluate on AIME 2024 [Mathematical Association of America, 2024], AIME 2025 [Mathematical Association of America, 2025], AMC [Mathematical Association of America, 2023], Math500 [Hendrycks et al., 2021], and Minerva Math [Lewkowycz et al., 2022], and compare against sequential baselines trained from the same base model and data source as well as external parallel and efficient-reasoning systems. We report final-answer accuracy and executable latency. For Parason, **model latency** is the token length of the longest generation path: sequential tokens are counted normally, while each `<Parallel>` block contributes the maximum branch length rather than the sum of all branches. This metric captures the wall-clock benefit available to an inference engine that runs branches concurrently.

4.1 Main Results

Parason improves accuracy over prior parallel reasoning systems. Table 2 compares Parason with existing parallel and efficient-reasoning methods. Parason gives the best reported AIME25 and AMC results among the listed systems, reaches the best Parason four-benchmark average accuracy of 84.7%, and stays close to ThreadWeaver on AIME24. Across the original settings and the additional $\alpha = 0.1$ runs, PA-GRPO reaches 70.6% on AIME25, 97.5% on AMC, and 94.6% on Math500. These gains show that Parason improves both accuracy and average benchmark coverage over prior parallel work, even though it uses an 8B model rather than several 32B baselines.

Parason gives a better token-latency–accuracy trade-off under constraints. Table 3 shows that Parason is most useful when the model has a small thinking budget on the longest token path. At 2048 tokens, the best Subtask-aware PA-GRPO model reaches 34.7% AIME24 accuracy, compared with 16.8% for SFT only. At 8192 tokens, it reaches 60.3%, compared with 41.8% for SFT only

Model	AIME24 Accuracy (%)			
	$B = 2{,}048$	$B = 8{,}192$	$B = 16{,}384$	$B = 24{,}576$
SFT only	16.8	41.8	66.8	70.2
PA-GRPO ($\beta_{\text{subtask}} = 0.025$)	$16.7_{-0.10}$	$45.2_{+3.40}$	$68.0_{+1.20}$	$73.2_{+3.00}$
PA-GRPO ($\beta_{\text{subtask}} = 0.050$)	$26.8_{+10.00}$	$60.0_{+18.20}$	$72.2_{+5.40}$	$74.8_{+4.60}$
PA-GRPO ($\beta_{\text{subtask}} = 0.100$)	$\mathbf{34.7}_{+17.90}$	$\mathbf{60.3}_{+18.50}$	$\mathbf{74.4}_{+7.60}$	$\mathbf{75.7}_{+5.50}$

Table 3: **Accuracy under token-latency budgets.** Each column reports AIME24 accuracy for a fixed thinking budget B on the longest token path. Subscripts show absolute accuracy-point gains relative to SFT only under the same budget; red indicates a decrease. Subtask-aware PA-GRPO provides the largest gains in low-budget regimes: at $B = 2{,}048$, $\beta_{\text{subtask}} = 0.100$ reaches 34.7% accuracy, compared with 16.8% for SFT only.

Method	Trigger Ratio	Token Latency	Acceleration Ratio	Avg. Math Acc.
ThreadWeaver (8B)	83.7	13.3k	1.18x	77.6
Multiverse (32B)	65.6	10.3k	1.16x	63.8
Parason-8B (w/ SFT only)	69.3	14.1k	1.27x	78.3
Original runs with $\alpha = 0.000$				
+ PA-GRPO ($\beta_{\text{trial}} = 0.025, \alpha = 0.000$)	77.3	15.2k	1.46x	79.9
+ PA-GRPO ($\beta_{\text{trial}} = 0.050, \alpha = 0.000$)	84.0	15.8k	1.61x	<u>80.4</u>
+ PA-GRPO ($\beta_{\text{trial}} = 0.100, \alpha = 0.000$)	**98.8**	15.7k	1.71x	**80.4**
+ PA-GRPO ($\beta_{\text{subtask}} = 0.025, \alpha = 0.000$)	70.1	14.9k	1.48x	79.5
+ PA-GRPO ($\beta_{\text{subtask}} = 0.050, \alpha = 0.000$)	79.3	16.6k	**1.75x**	80.3
+ PA-GRPO ($\beta_{\text{subtask}} = 0.100, \alpha = 0.000$)	89.6	15.4k	1.73x	80.1
Additional runs with $\alpha = 0.100$				
+ PA-GRPO ($\beta_{\text{trial}} = 0.025, \alpha = 0.100$)	62.3	**13.2k**	1.46x	79.4
+ PA-GRPO ($\beta_{\text{trial}} = 0.050, \alpha = 0.100$)	79.7	13.6k	1.50x	79.3
+ PA-GRPO ($\beta_{\text{trial}} = 0.100, \alpha = 0.100$)	95.3	14.5k	1.72x	80.3
+ PA-GRPO ($\beta_{\text{subtask}} = 0.025, \alpha = 0.100$)	53.7	<u>13.4k</u>	1.34x	79.0
+ PA-GRPO ($\beta_{\text{subtask}} = 0.050, \alpha = 0.100$)	81.3	**13.2k**	1.60x	78.0
+ PA-GRPO ($\beta_{\text{subtask}} = 0.100, \alpha = 0.100$)	<u>96.5</u>	14.3k	<u>1.74x</u>	78.0

Table 4: **PA-GRPO produces usable parallelism across latency-penalty settings.** For PA-GRPO, Trigger Ratio and Avg. Math Acc. are averaged over AIME24, AIME25, and Math500. **Acceleration ratio** is total tokens / longest-path tokens, and **trigger ratio** is the fraction of samples containing at least one `<Parallel>` block. Bold and underlined values mark the best and second-best PA-GRPO results among available measurements.

at the same budget. Thus, Parason does not simply spend more tokens; it moves useful work into parallel branches and keeps the longest path short.

4.2 Insights and Findings

Trial rewards give the clearest accuracy gains, while the latency penalty shortens the critical path. Table 2 compares PA-GRPO with ThreadWeaver, which obtains 81.0% average accuracy with a four-benchmark token latency of 14.8k. The original $\beta_{\text{trial}} = 0.100$ setting gives the best average accuracy (84.7%), while $\beta_{\text{trial}} = 0.050$ gives the best AIME25 result (70.6%). The best AIME24 result (78.2%) is shared by the original and additional $\beta_{\text{trial}} = 0.100$ settings, and the best AMC result (97.5%) is shared by the original $\beta_{\text{trial}} = 0.100$ model and the additional $\alpha = 0.1$, $\beta_{\text{subtask}} = 0.100$ model. Among the additional runs, $\beta_{\text{trial}} = 0.025$ gives the lowest four-benchmark token latency at 12.1k. The additional sweep spans 12.1–13.4k, below ThreadWeaver's 14.8k average latency.

PA-GRPO turns more parallel structure into executable acceleration. Table 4 shows that the learned parallel regions are usable by the runtime. SFT alone already triggers parallel execution on 69.3% of samples and yields 1.27× token-level acceleration. Under the original $\alpha = 0.000$ setting, the Trial sweep raises the trigger ratio from 77.3% to 98.8% and the acceleration ratio from 1.46× to 1.71×, while maintaining 79.9–80.4% average math accuracy. Subtask incentives

provide another route to high acceleration, with $\beta_{\text{subtask}} = 0.050$ reaching the best acceleration ratio of $1.75\times$ and 80.3% average accuracy. In the additional $\alpha = 0.100$ runs, acceleration rises with the parallelism coefficient: $\beta_{\text{subtask}} = 0.100$ reaches $1.74\times$ and $\beta_{\text{trial}} = 0.100$ reaches $1.72\times$. The lowest three-benchmark token latency is 13.2k, reached by $\beta_{\text{trial}} = 0.025$ and $\beta_{\text{subtask}} = 0.050$ after rounding.

	Token-Level Metrics					Wall-Clock Metrics	
Difficulty	#Problems	Generated #Tokens	Latency #Tokens	Acceleration Ratio	Saved #Tokens	Parallel (s)	Sequential (s)
Easy	13	21.3k	12.5k	$1.70\times$	8.8k	188.3	304.2
Medium	11	36.8k	21.2k	$1.74\times$	15.6k	336.2	464.7
Hard	6	50.3k	29.0k	$1.73\times$	21.3k	487.3	716.3

Table 5: **Harder questions save more tokens, while parallel execution reduces wall-clock latency across all difficulty levels.** We break down AIME24 thinking tokens and measured wall-clock time by difficulty. Latency #Tokens denotes the length of the longest token path, Acceleration Ratio is generated tokens divided by latency tokens, and Saved #Tokens is generated tokens minus latency tokens. The experiment is executed on A800 GPUs.

Harder questions create more parallelizable work, not less acceleration. Table 5 breaks down the step-200 AIME24 run by difficulty. Generated tokens grow sharply with difficulty, from 21.3k on easy problems to 36.8k on medium problems and 50.3k on hard problems. The longest token path also grows, but much more slowly: 12.5k, 21.2k, and 29.0k tokens, respectively. As a result, Parason keeps a stable 1.70–$1.74\times$ token-level acceleration ratio across all three difficulty groups. This translates into measured wall-clock speedups of $1.62\times$ on easy, $1.38\times$ on medium, and $1.47\times$ on hard problems. The absolute number of saved tokens increases from 8.8k on easy problems to 21.3k on hard problems, showing that harder questions expose more branch-level work that can be moved off the critical path. This supports the central motivation of Parason: difficult reasoning still requires long and diverse computation, but much of that computation does not need to remain serial.

5 Conclusion

In this work, we introduced **Parason**, an algorithm-system co-design framework that brings parallel processing to sequential LLM reasoning. Parason distinguishes deterministic **Subtask Parallelism** from speculative **Trial Parallelism**, and provides a stable pipeline for data curation, CFG-based structure, PA-GRPO training, and inference-engine execution. This design enables efficient parallel reasoning and improves the responsiveness of interactive agents, coding assistants, and scientific problem-solving tools. Our findings show that Trial Parallelism is a key mechanism for complex mathematical reasoning, although prior work has mainly focused on subtask decomposition. Across benchmarks, Parason reduces **token latency by about** $1.7\times$ while **maintaining comparable accuracy**. We hope Parason raises the community's awareness of parallel reasoning and its two parallelism schemes, and inspires future work on more efficient and scalable reasoning systems. We include limitations and the impact statement in Appendix A, and will open-source our implementation to support reproducibility.

Acknowledgments and Disclosure of Funding

References

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025. URL `https://hkunlp.github.io/blog/2025/Polaris`.

Anthropic. Claude code: Deep coding at terminal velocity. `https://www.anthropic.com/claude-code`, 2025a.

Anthropic. Introducing claude opus 4.5. `https://www.anthropic.com/news/claude-opus-4-5`, 2025b.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Nuelle, Niklas Hoffra, et al. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*, 2023.

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

Google DeepMind. Ai solves imo problems at silver medal level. https://deepmind.google/blog/ai-solves-imo-problems-at-silver-medal-level/, 2024. Accessed: 2024-07-25.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf.

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

Yixin Dong, Charlie F Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. Xgrammar: Flexible and efficient structured generation engine for large language models. *arXiv preprint arXiv:2411.15100*, 2024. URL https://arxiv.org/abs/2411.15100.

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

Google. Gemini 2.5: Our most intelligent ai model, March 2025. URL https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/#gemini-2-5-thinking.

Google DeepMind. Introducing gemini 3 pro. https://deepmind.google/technologies/gemini/, 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, 2021. URL https://arxiv.org/abs/2103.03874.

Jingcheng Hu, Yinmin Zhang, Shijie Shang, Xiaobo Yang, Yue Peng, Zhewei Huang, Hebin Zhou, Xin Wu, Jie Cheng, Fanqi Wan, Xiangwen Kong, Chengyuan Yao, Kaiwen Yan, Ailin Huang, Hongyu Zhou, Qi Han, Zheng Ge, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Pacore: Learning to scale test-time compute with parallel coordinated reasoning, 2026. URL https://arxiv.org/abs/2601.05593.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, Ottavia Bertolli, Tom Zahavy, Amol Mandhane, Jessica Yung, Iuliya Beloshapka, Borja Ibarz, Vivek Veeriah, Lei Yu, Oliver Nash, Paul Lezeau, Salvatore Mercuri, Calle Sönne, Bhavik Mehta, Alex Davies, Daniel Zheng, Fabian Pedregosa, Yin Li, Ingrid von Glehn, Mark Rowland, Samuel Albanie, Ameya Velingker, Simon Schmitt, Edward Lockhart, Edward Hughes, Henryk Michalewski, Nicolas Sonnerat, Demis Hassabis, Pushmeet Kohli, and David Silver. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, November 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09833-y. URL http://dx.doi.org/10.1038/s41586-025-09833-y.

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*, 2025.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022. URL https://arxiv.org/abs/2206.14858.

Long Lian, Sida Wang, Felix Juefei-Xu, Tsu-Jui Fu, Xiuyu Li, Adam Yala, Trevor Darrell, Alane Suhr, Yuandong Tian, and Xi Victoria Lin. Threadweaver: Adaptive threading for efficient parallel reasoning in language models. *arXiv preprint arXiv:2501.00000*, 2025.

Thang Luong, Dawsen Hwang, Hoang H. Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Clara Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu H. Trinh, Quoc V. Le, and Junehyuk Jung. Towards robust mathematical reasoning, 2025. URL https://arxiv.org/abs/2511.01846.

Mathematical Association of America. American Mathematics Competitions 2023, 2023. URL https://maa.org/student-programs/amc/. Accessed: 2026-05-07.

Mathematical Association of America. American Invitational Mathematics Examination 2024, 2024. URL https://artofproblemsolving.com/wiki/index.php/American_Invitational_Mathematics_Examination. Accessed: 2025-05-14.

Mathematical Association of America. American Invitational Mathematics Examination 2025, 2025. URL https://artofproblemsolving.com/wiki/index.php/American_Invitational_Mathematics_Examination. Accessed: 2025-05-14.

MiniMax AI. Minimax m2: Open weight large language model. https://www.minimax.io/news/minimax-m2, 2025.

Moonshot AI. Kimi k2: Open agentic intelligence. https://moonshotai.github.io/Kimi-K2/, 2025.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset, 2025. URL https://arxiv.org/abs/2504.16891.

Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. Skeleton-of-thought: Large language models can do parallel decoding. *Proceedings ENLSP-III*, 2023.

OpenAI. Codex: Ai coding partner from openai. https://openai.com/codex, 2025.

OpenAI. Gpt-5.5. https://openai.com/, 2026.

OpenAI, Lama Ahmad, Amanda Askell, Pamela Mishkin, Thomas O'Keefe, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024. URL https://arxiv.org/abs/2412.16720.

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.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, Michael Choi, Anish Agrawal, Arnav Chopra, Adam Khoja, Ryan Kim, Richard Ren, Jason Hausenloy, Oliver Zhang, Mantas Mazeika, Dmitry Dodonov, Tung Nguyen, Jaeho Lee, Daron Anderson, Mikhail Doroshenko, Alun Cennyth Stokes, Mobeen Mahmood, Oleksandr Pokutnyi, Oleg Iskra, Jessica P. Wang, John-Clark Levin, Mstyslav Kazakov, Fiona Feng, Steven Y. Feng, Haoran Zhao, Michael Yu, Varun Gangal, Chelsea Zou, Zihan Wang, Serguei Popov, Robert Gerbicz, Geoff Galgon, Johannes Schmitt, Will Yeadon, Yongki Lee, Scott Sauers, Alvaro Sanchez, Fabian Giska, Marc Roth, Søren Riis, Saiteja Utpala, Noah Burns, Gashaw M. Goshu, Mohinder Maheshbhai Naiya, Chidozie Agu, Zachary Giboney, Antrell Cheatom, Francesco Fournier-Facio, Sarah-Jane Crowson, Lennart Finke, Zerui Cheng, Jennifer Zampese, Ryan G. Hoerr, Mark Nandor, Hyunwoo Park, Tim Gehrunger, Jiaqi Cai, Ben McCarty, Alexis C Garretson, Edwin Taylor, Damien Sileo, Qiuyu Ren, Usman Qazi, Lianghui Li, Jungbae Nam, John B. Wydallis, Pavel Arkhipov, Jack Wei Lun Shi, Aras Bacho, Chris G. Willcocks, Hangrui Cao, Sumeet Motwani, Emily de Oliveira Santos, Johannes Veith, Edward Vendrow, Doru Cojoc, Kengo Zenitani, Joshua Robinson, Longke Tang, Yuqi Li, Joshua Vendrow, Natanael Wildner Fraga, Vladyslav Kuchkin, Andrey Pupasov Maksimov, Pierre Marion, Denis Efremov, Jayson Lynch, Kaiqu Liang, Aleksandar Mikov, Andrew Gritsevskiy, Julien Guillod, Gözdenur Demir, Dakotah Martinez, Ben Pageler, Kevin Zhou, Saeed Soori, Ori Press, Henry Tang, Paolo Rissone, Sean R. Green, Lina Brüssel, Moon Twayana, Aymeric Dieuleveut, Joseph Marvin Imperial, Ameya Prabhu, Jinzhou Yang, Nick Crispino, Arun Rao, Dimitri Zvonkine, Gabriel Loiseau, Mikhail Kalinin, Marco Lukas, Ciprian Manolescu, Nate Stambaugh, Subrata Mishra, Tad Hogg, Carlo Bosio, Brian P Coppola, Julian Salazar, Jaehyeok Jin, Rafael Sayous, Stefan Ivanov, Philippe Schwaller, Shaipranesh Senthilkuma, Andres M Bran, Andres Algaba, Kelsey Van den Houte, Lynn Van Der Sypt, Brecht Verbeken, David Noever, Alexei Kopylov, Benjamin Myklebust, Bikun Li, Lisa Schut, Evgenii Zheltonozhskii, Qiaochu Yuan, Derek Lim, Richard Stanley, Tong Yang, John Maar, Julian Wykowski, Martí Oller, Anmol Sahu, Cesare Giulio Ardito, Yuzheng Hu, Ariel Ghislain Kemogne Kamdoum, Alvin Jin, Tobias Garcia Vilchis, Yuexuan Zu, Martin Lackner, James Koppel, Gongbo Sun, Daniil S. Antonenko, Steffi Chern, Bingchen Zhao, Pierrot Arsene, Joseph M Cavanagh, Daofeng Li, Jiawei Shen, Donato Crisostomi, Wenjin Zhang, Ali Dehghan, Sergey Ivanov, David Perrella, Nurdin Kaparov, Allen Zang, Ilia Sucholutsky, Arina Kharlamova, Daniil Orel, Vladislav Poritski, Shalev Ben-David, Zachary Berger, Parker Whitfill, Michael Foster, Daniel Munro, Linh Ho, Shankar Sivarajan, Dan Bar Hava, Aleksey Kuchkin, David Holmes, Alexandra Rodriguez-Romero, Frank Sommerhage, Anji Zhang, Richard Moat, Keith Schneider, Zakayo Kazibwe, Don Clarke, Dae Hyun Kim, Felipe Meneguitti Dias, Sara Fish, Veit Elser, Tobias Kreiman, Victor Efren Guadarrama Vilchis, Immo Klose, Ujjwala Anantheswaran, Adam Zweiger, Kaivalya Rawal, Jeffery Li, Jeremy Nguyen, Nicolas Daans, Haline Heidinger, Maksim Radionov, Václav Rozhoň, Vincent Ginis, Christian Stump, Niv Cohen, Rafał Poświata, Josef Tkadlec, Alan Goldfarb, Chenguang Wang, Piotr Padlewski, Stanislaw Barzowski, Kyle Montgomery, Ryan Stendall, Jamie Tucker-Foltz, Jack Stade, T. Ryan Rogers, Tom Goertzen, Declan Grabb, Abhishek Shukla, Alan Givré, John Arnold Ambay, Archan Sen, Muhammad Fayez Aziz, Mark H Inlow, Hao He, Ling Zhang, Younesse Kaddar, Ivar Ängquist, Yanxu Chen, Harrison K Wang, Kalyan Ramakrishnan, Elliott Thornley, Antonio Terpin, Hailey Schoelkopf, Eric Zheng, Avishy Carmi, Ethan D. L. Brown, Kelin Zhu, Max Bartolo, Richard Wheeler, Martin Stehberger, Peter Bradshaw, JP Heimonen, Kaustubh Sridhar, Ido Akov, Jennifer Sandlin, Yury Makarychev, Joanna Tam, Hieu Hoang, David M. Cunningham, Vladimir Goryachev, Demosthenes Patramanis, Michael Krause, Andrew Redenti, David Aldous, Jesyin Lai, Shannon Coleman, Jiangnan Xu, Sangwon Lee, Ilias Magoulas, Sandy Zhao, Ning Tang, Michael K. Cohen, Orr Paradise, Jan Hendrik Kirchner, Maksym Ovchynnikov, Jason O. Matos, Adithya Shenoy, Michael Wang, Yuzhou Nie, Anna Sztyber-Betley, Paolo Faraboschi, Robin Riblet, Jonathan Crozier, Shiv Halasyamani, Shreyas Verma, Prashant Joshi, Eli Meril, Ziqiao Ma, Jérémy Andréoletti, Raghav Singhal, Jacob Platnick, Volodymyr Nevirkovets, Luke Basler, Alexander Ivanov, Seri Khoury, Nils Gustafsson,

Marco Piccardo, Hamid Mostaghimi, Qijia Chen, Virendra Singh, Tran Quoc Khánh, Paul Rosu, Hannah Szlyk, Zachary Brown, Himanshu Narayan, Aline Menezes, Jonathan Roberts, William Alley, Kunyang Sun, Arkil Patel, Max Lamparth, Anka Reuel, Linwei Xin, Hanmeng Xu, Jacob Loader, Freddie Martin, Zixuan Wang, Andrea Achilleos, Thomas Preu, Tomek Korbak, Ida Bosio, Fereshteh Kazemi, Ziye Chen, Biró Bálint, Eve J. Y. Lo, Jiaqi Wang, Maria Inês S. Nunes, Jeremiah Milbauer, M Saiful Bari, Zihao Wang, Behzad Ansarinejad, Yewen Sun, Stephane Durand, Hossam Elgnainy, Guillaume Douville, Daniel Tordera, George Balabanian, Hew Wolff, Lynna Kvistad, Hsiaoyun Milliron, Ahmad Sakor, Murat Eron, Andrew Favre D. O., Shailesh Shah, Xiaoxiang Zhou, Firuz Kamalov, Sherwin Abdoli, Tim Santens, Shaul Barkan, Allison Tee, Robin Zhang, Alessandro Tomasiello, G. Bruno De Luca, Shi-Zhuo Looi, Vinh-Kha Le, Noam Kolt, Jiayi Pan, Emma Rodman, Jacob Drori, Carl J Fossum, Niklas Muennighoff, Milind Jagota, Ronak Pradeep, Honglu Fan, Jonathan Eicher, Michael Chen, Kushal Thaman, William Merrill, Moritz Firsching, Carter Harris, Stefan Ciobâcă, Jason Gross, Rohan Pandey, Ilya Gusev, Adam Jones, Shashank Agnihotri, Pavel Zhelnov, Mohammadreza Mofayezi, Alexander Piperski, David K. Zhang, Kostiantyn Dobarskyi, Roman Leventov, Ignat Soroko, Joshua Duersch, Vage Taamazyan, Andrew Ho, Wenjie Ma, William Held, Ruicheng Xian, Armel Randy Zebaze, Mohanad Mohamed, Julian Noah Leser, Michelle X Yuan, Laila Yacar, Johannes Lengler, Katarzyna Olszewska, Claudio Di Fratta, Edson Oliveira, Joseph W. Jackson, Andy Zou, Muthu Chidambaram, Timothy Manik, Hector Haffenden, Dashiell Stander, Ali Dasouqi, Alexander Shen, Bita Golshani, David Stap, Egor Kretov, Mikalai Uzhou, Alina Borisovna Zhidkovskaya, Nick Winter, Miguel Orbegozo Rodriguez, Robert Lauff, Dustin Wehr, Colin Tang, Zaki Hossain, Shaun Phillips, Fortuna Samuele, Fredrik Ekström, Angela Hammon, Oam Patel, Faraz Farhidi, George Medley, Forough Mohammadzadeh, Madellene Peñaflor, Haile Kassahun, Alena Friedrich, Rayner Hernandez Perez, Daniel Pyda, Taom Sakal, Omkar Dhamane, Ali Khajegili Mirabadi, Eric Hallman, Kenchi Okutsu, Mike Battaglia, Mohammad Maghsoudimehrabani, Alon Amit, Dave Hulbert, Roberto Pereira, Simon Weber, Handoko, Anton Peristyy, Stephen Malina, Mustafa Mehkary, Rami Aly, Frank Reidegeld, Anna-Katharina Dick, Cary Friday, Mukhwinder Singh, Hassan Shapourian, Wanyoung Kim, Mariana Costa, Hubeyb Gurdogan, Harsh Kumar, Chiara Ceconello, Chao Zhuang, Haon Park, Micah Carroll, Andrew R. Tawfeek, Stefan Steinerberger, Daattavya Aggarwal, Michael Kirchhof, Linjie Dai, Evan Kim, Johan Ferret, Jainam Shah, Yuzhou Wang, Minghao Yan, Krzysztof Burdzy, Lixin Zhang, Antonio Franca, Diana T. Pham, Kang Yong Loh, Joshua Robinson, Abram Jackson, Paolo Giordano, Philipp Petersen, Adrian Cosma, Jesus Colino, Colin White, Jacob Votava, Vladimir Vinnikov, Ethan Delaney, Petr Spelda, Vit Stritecky, Syed M. Shahid, Jean-Christophe Mourrat, Lavr Vetoshkin, Koen Sponselee, Renas Bacho, Zheng-Xin Yong, Florencia de la Rosa, Nathan Cho, Xiuyu Li, Guillaume Malod, Orion Weller, Guglielmo Albani, Leon Lang, Julien Laurendeau, Dmitry Kazakov, Fatimah Adesanya, Julien Portier, Lawrence Hollom, Victor Souza, Yuchen Anna Zhou, Julien Degorre, Yiğit Yalın, Gbenga Daniel Obikoya, Rai, Filippo Bigi, M. C. Boscá, Oleg Shumar, Kaniuar Bacho, Gabriel Recchia, Mara Popescu, Nikita Shulga, Ngefor Mildred Tanwie, Thomas C. H. Lux, Ben Rank, Colin Ni, Matthew Brooks, Alesia Yakimchyk, Huanxu, Liu, Stefano Cavalleri, Olle Häggström, Emil Verkama, Joshua Newbould, Hans Gundlach, Leonor Brito-Santana, Brian Amaro, Vivek Vajipey, Rynaa Grover, Ting Wang, Yosi Kratish, Wen-Ding Li, Sivakanth Gopi, Andrea Caciolai, Christian Schroeder de Witt, Pablo Hernández-Cámara, Emanuele Rodolà, Jules Robins, Dominic Williamson, Vincent Cheng, Brad Raynor, Hao Qi, Ben Segev, Jingxuan Fan, Sarah Martinson, Erik Y. Wang, Kaylie Hausknecht, Michael P. Brenner, Mao Mao, Christoph Demian, Peyman Kassani, Xinyu Zhang, David Avagian, Eshawn Jessica Scipio, Alon Ragoler, Justin Tan, Blake Sims, Rebeka Plecnik, Aaron Kirtland, Omer Faruk Bodur, D. P. Shinde, Yan Carlos Leyva Labrador, Zahra Adoul, Mohamed Zekry, Ali Karakoc, Tania C. B. Santos, Samir Shamseldeen, Loukmane Karim, Anna Liakhovitskaia, Nate Resman, Nicholas Farina, Juan Carlos Gonzalez, Gabe Maayan, Earth Anderson, Rodrigo De Oliveira Pena, Elizabeth Kelley, Hodjat Mariji, Rasoul Pouriamanesh, Wentao Wu, Ross Finocchio, Ismail Alarab, Joshua Cole, Danyelle Ferreira, Bryan Johnson, Mohammad Safdari, Liangti Dai, Siriphan Arthornthurasuk, Isaac C. McAlister, Alejandro José Moyano, Alexey Pronin, Jing Fan, Angel Ramirez-Trinidad, Yana Malysheva, Daphiny Pottmaier, Omid Taheri, Stanley Stepanic, Samuel Perry, Luke Askew, Raúl Adrián Huerta Rodríguez, Ali M. R. Minissi, Ricardo Lorena, Krishnamurthy Iyer, Arshad Anil Fasiludeen, Ronald Clark, Josh Ducey, Matheus Piza, Maja Somrak, Eric Vergo, Juehang Qin, Benjámin Borbás, Eric Chu, Jack Lindsey, Antoine Jallon, I. M. J. McInnis, Evan Chen, Avi Semler, Luk Gloor, Tej Shah, Marc Carauleanu, Pascal Lauer, Tran Đuc Huy, Hossein Shahrtash, Emilien Duc, Lukas Lewark, Assaf Brown, Samuel Albanie, Brian Weber, Warren S. Vaz, Pierre Clavier, Yiyang Fan, Gabriel Poesia Reis e Silva, Long, Lian, Marcus

Abramovitch, Xi Jiang, Sandra Mendoza, Murat Islam, Juan Gonzalez, Vasilios Mavroudis, Justin Xu, Pawan Kumar, Laxman Prasad Goswami, Daniel Bugas, Nasser Heydari, Ferenc Jeanplong, Thorben Jansen, Antonella Pinto, Archimedes Apronti, Abdallah Galal, Ng Ze-An, Ankit Singh, Tong Jiang, Joan of Arc Xavier, Kanu Priya Agarwal, Mohammed Berkani, Gang Zhang, Zhehang Du, Benedito Alves de Oliveira Junior, Dmitry Malishev, Nicolas Remy, Taylor D. Hartman, Tim Tarver, Stephen Mensah, Gautier Abou Loume, Wiktor Morak, Farzad Habibi, Sarah Hoback, Will Cai, Javier Gimenez, Roselynn Grace Montecillo, Jakub Łucki, Russell Campbell, Asankhaya Sharma, Khalida Meer, Shreen Gul, Daniel Espinosa Gonzalez, Xavier Alapont, Alex Hoover, Gunjan Chhablani, Freddie Vargus, Arunim Agarwal, Yibo Jiang, Deepakkumar Patil, David Outevsky, Kevin Joseph Scaria, Rajat Maheshwari, Abdelkader Dendane, Priti Shukla, Ashley Cartwright, Sergei Bogdanov, Niels Mündler, Sören Möller, Luca Arnaboldi, Kunvar Thaman, Muhammad Rehan Siddiqi, Prajvi Saxena, Himanshu Gupta, Tony Fruhauff, Glen Sherman, Mátyás Vincze, Siranut Usawasutsakorn, Dylan Ler, Anil Radhakrishnan, Innocent Enyekwe, Sk Md Salauddin, Jiang Muzhen, Aleksandr Maksapetyan, Vivien Rossbach, Chris Harjadi, Mohsen Bahaloohoreh, Claire Sparrow, Jasdeep Sidhu, Sam Ali, Song Bian, John Lai, Eric Singer, Justine Leon Uro, Greg Bateman, Mohamed Sayed, Ahmed Menshawy, Darling Duclosel, Dario Bezzi, Yashaswini Jain, Ashley Aaron, Murat Tiryakioglu, Sheeshram Siddh, Keith Krenek, Imad Ali Shah, Jun Jin, Scott Creighton, Denis Peskoff, Zienab EL-Wasif, Ragavendran P V, Michael Richmond, Joseph McGowan, Tejal Patwardhan, Hao-Yu Sun, Ting Sun, Nikola Zubić, Samuele Sala, Stephen Ebert, Jean Kaddour, Manuel Schottdorf, Dianzhuo Wang, Gerol Petruzella, Alex Meiburg, Tilen Medved, Ali ElSheikh, S Ashwin Hebbar, Lorenzo Vaquero, Xianjun Yang, Jason Poulos, Vilém Zouhar, Sergey Bogdanik, Mingfang Zhang, Jorge Sanz-Ros, David Anugraha, Yinwei Dai, Anh N. Nhu, Xue Wang, Ali Anil Demircali, Zhibai Jia, Yuyin Zhou, Juncheng Wu, Mike He, Nitin Chandok, Aarush Sinha, Gaoxiang Luo, Long Le, Mickaël Noyé, Michał Perełkiewicz, Ioannis Pantidis, Tianbo Qi, Soham Sachin Purohit, Letitia Parcalabescu, Thai-Hoa Nguyen, Genta Indra Winata, Edoardo M. Ponti, Hanchen Li, Kaustubh Dhole, Jongee Park, Dario Abbondanza, Yuanli Wang, Anupam Nayak, Diogo M. Caetano, Antonio A. W. L. Wong, Maria del Rio-Chanona, Dániel Kondor, Pieter Francois, Ed Chalstrey, Jakob Zsambok, Dan Hoyer, Jenny Reddish, Jakob Hauser, Francisco-Javier Rodrigo-Ginés, Suchandra Datta, Maxwell Shepherd, Thom Kamphuis, Qizheng Zhang, Hyunjun Kim, Ruiji Sun, Jianzhu Yao, Franck Dernoncourt, Satyapriya Krishna, Sina Rismanchian, Bonan Pu, Francesco Pinto, Yingheng Wang, Kumar Shridhar, Kalon J. Overholt, Glib Briia, Hieu Nguyen, David, Soler Bartomeu, Tony CY Pang, Adam Wecker, Yifan Xiong, Fanfei Li, Lukas S. Huber, Joshua Jaeger, Romano De Maddalena, Xing Han Lù, Yuhui Zhang, Claas Beger, Patrick Tser Jern Kon, Sean Li, Vivek Sanker, Ming Yin, Yihao Liang, Xinlu Zhang, Ankit Agrawal, Li S. Yifei, Zechen Zhang, Mu Cai, Yasin Sonmez, Costin Cozianu, Changhao Li, Alex Slen, Shoubin Yu, Hyun Kyu Park, Gabriele Sarti, Marcin Briański, Alessandro Stolfo, Truong An Nguyen, Mike Zhang, Yotam Perlitz, Jose Hernandez-Orallo, Runjia Li, Amin Shabani, Felix Juefei-Xu, Shikhar Dhingra, Orr Zohar, My Chiffon Nguyen, Alexander Pondaven, Abdurrahim Yilmaz, Xuandong Zhao, Chuanyang Jin, Muyan Jiang, Stefan Todoran, Xinyao Han, Jules Kreuer, Brian Rabern, Anna Plassart, Martino Maggetti, Luther Yap, Robert Geirhos, Jonathon Kean, Dingsu Wang, Sina Mollaei, Chenkai Sun, Yifan Yin, Shiqi Wang, Rui Li, Yaowen Chang, Anjiang Wei, Alice Bizeul, Xiaohan Wang, Alexandre Oliveira Arrais, Kushin Mukherjee, Jorge Chamorro-Padial, Jiachen Liu, Xingyu Qu, Junyi Guan, Adam Bouyamourn, Shuyu Wu, Martyna Plomecka, Junda Chen, Mengze Tang, Jiaqi Deng, Shreyas Subramanian, Haocheng Xi, Haoxuan Chen, Weizhi Zhang, Yinuo Ren, Haoqin Tu, Sejong Kim, Yushun Chen, Sara Vera Marjanović, Junwoo Ha, Grzegorz Luczyna, Jeff J. Ma, Zewen Shen, Dawn Song, Cedegao E. Zhang, Zhun Wang, Gaël Gendron, Yunze Xiao, Leo Smucker, Erica Weng, Kwok Hao Lee, Zhe Ye, Stefano Ermon, Ignacio D. Lopez-Miguel, Theo Knights, Anthony Gitter, Namkyu Park, Boyi Wei, Hongzheng Chen, Kunal Pai, Ahmed Elkhanany, Han Lin, Philipp D. Siedler, Jichao Fang, Ritwik Mishra, Károly Zsolnai-Fehér, Xilin Jiang, Shadab Khan, Jun Yuan, Rishab Kumar Jain, Xi Lin, Mike Peterson, Zhe Wang, Aditya Malusare, Maosen Tang, Isha Gupta, Ivan Fosin, Timothy Kang, Barbara Dworakowska, Kazuki Matsumoto, Guangyao Zheng, Gerben Sewuster, Jorge Pretel Villanueva, Ivan Rannev, Igor Chernyavsky, Jiale Chen, Deepayan Banik, Ben Racz, Wenchao Dong, Jianxin Wang, Laila Bashmal, Duarte V. Gonçalves, Wei Hu, Kaushik Bar, Ondrej Bohdal, Atharv Singh Patlan, Shehzaad Dhuliawala, Caroline Geirhos, Julien Wist, Yuval Kansal, Bingsen Chen, Kutay Tire, Atak Talay Yücel, Brandon Christof, Veerupaksh Singla, Zijian Song, Sanxing Chen, Jiaxin Ge, Kaustubh Ponkshe, Isaac Park, Tianneng Shi, Martin Q. Ma, Joshua Mak, Sherwin Lai, Antoine Moulin, Zhuo Cheng, Zhanda Zhu, Ziyi Zhang, Vaidehi Patil, Ketan Jha, Qiutong Men, Jiaxuan Wu, Tianchi Zhang, Bruno Hebling Vieira, Alham Fikri Aji, Jae-Won Chung,

Mohammed Mahfoud, Ha Thi Hoang, Marc Sperzel, Wei Hao, Kristof Meding, Sihan Xu, Vassilis Kostakos, Davide Manini, Yueying Liu, Christopher Toukmaji, Jay Paek, Eunmi Yu, Arif Engin Demircali, Zhiyi Sun, Ivan Dewerpe, Hongsen Qin, Roman Pflugfelder, James Bailey, Johnathan Morris, Ville Heilala, Sybille Rosset, Zishun Yu, Peter E. Chen, Woongyeong Yeo, Eeshaan Jain, Ryan Yang, Sreekar Chigurupati, Julia Chernyavsky, Sai Prajwal Reddy, Subhashini Venugopalan, Hunar Batra, Core Francisco Park, Hieu Tran, Guilherme Maximiano, Genghan Zhang, Yizhuo Liang, Hu Shiyu, Rongwu Xu, Rui Pan, Siddharth Suresh, Ziqi Liu, Samaksh Gulati, Songyang Zhang, Peter Turchin, Christopher W. Bartlett, Christopher R. Scotese, Phuong M. Cao, Ben Wu, Jacek Karwowski, Davide Scaramuzza, Aakaash Nattanmai, Gordon McKellips, Anish Cheraku, Asim Suhail, Ethan Luo, Marvin Deng, Jason Luo, Ashley Zhang, Kavin Jindel, Jay Paek, Kasper Halevy, Allen Baranov, Michael Liu, Advaith Avadhanam, David Zhang, Vincent Cheng, Brad Ma, Evan Fu, Liam Do, Joshua Lass, Hubert Yang, Surya Sunkari, Vishruth Bharath, Violet Ai, James Leung, Rishit Agrawal, Alan Zhou, Kevin Chen, Tejas Kalpathi, Ziqi Xu, Gavin Wang, Tyler Xiao, Erik Maung, Sam Lee, Ryan Yang, Roy Yue, Ben Zhao, Julia Yoon, Sunny Sun, Aryan Singh, Ethan Luo, Clark Peng, Tyler Osbey, Taozhi Wang, Daryl Echeazu, Hubert Yang, Timothy Wu, Spandan Patel, Vidhi Kulkarni, Vijaykaarti Sundarapandiyan, Ashley Zhang, Andrew Le, Zafir Nasim, Srikar Yalam, Ritesh Kasamsetty, Soham Samal, Hubert Yang, David Sun, Nihar Shah, Abhijeet Saha, Alex Zhang, Leon Nguyen, Laasya Nagumalli, Kaixin Wang, Alan Zhou, Aidan Wu, Jason Luo, Anwith Telluri, Summer Yue, Alexandr Wang, and Dan Hendrycks. Humanity's last exam, 2025. URL `https://arxiv.org/abs/2501.14249`.

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.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL `https://github.com/huggingface/trl`.

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.

Ziqi Wang, Boye Niu, Zipeng Gao, Zhi Zheng, Tong Xu, Linghui Meng, Zhongli Li, Jing Liu, Yilong Chen, Chen Zhu, et al. A survey on parallel reasoning. *arXiv preprint arXiv:2510.12164*, 2025.

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

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.

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.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs. *arXiv preprint arXiv:2312.07104*, 2023. URL `https://arxiv.org/abs/2312.07104`.

Tong Zheng, Hongming Zhang, Wenhao Yu, Xiaoyang Wang, Xinyu Yang, Runpeng Dai, Rui Liu, Huiwen Bao, Chengsong Huang, Heng Huang, et al. Parallel-r1: Towards parallel thinking via reinforcement learning. *arXiv preprint arXiv:2509.07980*, 2025.

A Limitations and Future Work

Parason's training and evaluation mainly focus on mathematical reasoning. It remains unclear how well the same taxonomy, data curation pipeline, and PA-GRPO objective transfer to other domains, such as real-world agents. Second, our current experiments focus on 8B-scale models. This setting provides a controlled testbed for studying parallel reasoning, but it does not fully show how Parason behaves across model families and sizes. In future work, we will scale Parason to more backbones and larger models to study whether the same parallelism patterns and latency gains hold at broader scales.

B Prompts Used for Data Curation

We use the following prompts to identify Trial and Subtask steps in LLM reasoning trajectories.

For open-source models, we first split each reasoning trajectory by `\n\n`. The first prompt reconstructs logical steps by deciding whether each segment starts a new subproblem or continues a previous one. The second prompt then classifies each logical step as either a Trial step or a Subtask step.

For commercial models, the released reasoning summaries already mark steps with "***Step i***". We therefore apply the second prompt directly to classify each marked step as Trial or Subtask.

Reconstruction by Logical Steps

SYSTEM:

```
You are a helpful assistant that analyzes chain-of-thought traces from
↪  reasoning models.

The chain of thought has been split into multiple steps. For each step, decide
↪  whether it starts a new subproblem or continues the previous one.

Use the content of each step to make the decision. If a step introduces a new
↪  concept, question, or task that is distinct from previous steps, it likely
↪  starts a new subproblem. If it builds on previous steps by adding details,
↪  explanations, reflections, or calculations about the same concept, it
↪  likely continues the previous subproblem.

More specifically, steps that start with "Alternatively", "Wait", or "But" are
↪  likely to start a new subproblem, while steps that start with "Therefore",
↪  "Thus", or "Consequently" are likely to continue the previous subproblem.
Your output should be in the following format:

Step i: [New Subproblem|Continue Previous Subproblem]

If a step continues a previous subproblem, report where that subproblem
↪  starts. For example, if Step 5 continues the subproblem that started at
↪  Step 3, output:
Step 5: Continue Previous Subproblem (started at Step 3)

IMPORTANT: Output the analysis in the specified format. Use one line per step.
↪  Do not include additional explanations or text.

Example:
Step 1: New Subproblem
Step 2: Continue Previous Subproblem (started at Step 1)
Step 3: New Subproblem
Step 4: Continue Previous Subproblem (started at Step 3)
Step 5: Continue Previous Subproblem (started at Step 3)
```

Parallel Stage Identification

SYSTEM:

```
You are given a series of mathematical reasoning steps. Classify each step
↪  into one of the following categories:
1. Trial Step: A step that introduces a new idea, approach, or line of thought
↪  whose success is uncertain but worth exploring.
2. Subtask Step: A step that belongs to a known solution path and directly
↪  contributes to the final solution.

Your output should be in the following format:
Step i: [Trial Step|Subtask Step] [Reason]

The number of output steps should match the number of input steps. Each step
↪  has its own number and is separated by "================================
↪  ================================================". Ignore the content under
↪  "Content after </think> tag:
"

Give the reason for each classification after the step type.
```

C AIME 2024 Problem Indices

Table 6 lists the AIME 2024 validation problems used for the difficulty-based analysis in Table 5. We group problems into Easy, Mid, and Hard subsets according to their difficulty labels, and report the corresponding zero-based problem indices for reproducibility.

Difficulty	Problem Indices
Hard	2, 3, 13, 21, 28, 29
Mid	1, 4, 5, 10, 16, 17, 18, 20, 25, 26, 27
Easy	0, 6, 7, 8, 9, 11, 12, 14, 15, 19, 22, 23, 24

Table 6: AIME 2024 Problem Classification by Difficulty.