METACTRL: YOUR LARGE LANGUAGE MODELS CAN REASON BETTER AND MORE CONCISELY WITH A METACOGNITIVE CONTROLLER

Anonymous authors
Paper under double-blind review

ABSTRACT

Large reasoning models improve performance on challenging problems by allocating additional computation before answering, but longer reasoning does not always lead to better results and can introduce substantial redundant reasoning on simple problems. Conversely, aggressively shortening reasoning can degrade performance on difficult ones. Effective reasoning therefore requires dynamically deciding when additional computation is useful based on the reasoner's capabilities and evolving solution state. Existing approaches often rely on predefined budgets or intervention rules, retrain the target reasoner, or require additional supervision. We introduce **MetaCtrl**, a lightweight controller that adaptively regulates a frozen reasoner without predefined token budgets or reasoner retraining. We formulate reasoning regulation as a sequential metacognitive control problem: MetaCtrl observes the evolving reasoning trace and decides whether to continue, simplify, skip redundant steps, or conclude reasoning. It is trained directly with reinforcement learning using a reward that prioritizes correctness while favoring shorter trajectories among correct solutions, requiring neither supervised intervention trajectories nor problem-specific budgets. Across seven benchmarks spanning mathematics, science, and code, MetaCtrl consistently improves the accuracy of LRMs while reducing their reasoning length. On DeepSeek-R1-Distill-Qwen-7B, it improves average accuracy by 4.7 points while reducing generation length by 53.3%. Without further training, the same controller transfers to an unseen reasoner (e.g., Qwen3-14B), improving average accuracy by 2.9 points and reducing generation length by 50.3%. These results establish MetaCtrl as a plug-and-play controller for improving reasoning accuracy while substantially reducing inference-time generation.

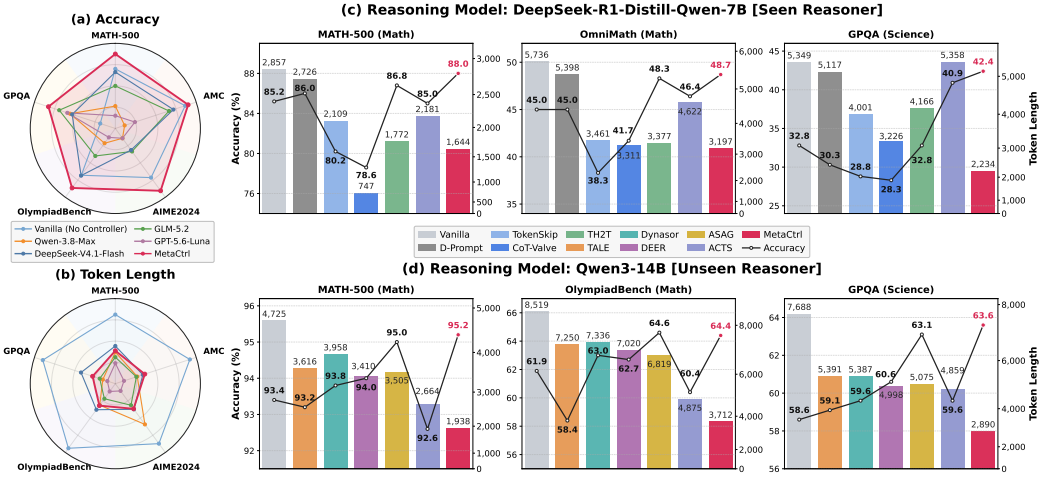


Figure 1: **Left:** Accuracy and generation length of the same reasoner when controlled by MetaCtrl, controlled by frontier LLMs, or run without external control across five benchmarks. **Right:** Comparison of MetaCtrl with representative efficient reasoning baselines across two reasoning models.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Recent large reasoning models (LRMs) (Jaech et al., 2024; Guo et al., 2025; Yang et al., 2025a) have shown that scaling test-time computation (Snell et al., 2025) through chain of thought (Kojima et al., 2022; Wei et al., 2022; Yao et al., 2023; Sprague et al., 2025) reasoning can substantially improve performance on challenging tasks (Phan et al., 2025; Li et al., 2025; Cao et al., 2025; Yu et al., 2025b; Chen et al., 2026). However, longer reasoning steps do no always lead to better performance. LRMs often spend excessive computation on redundant verification and unnecessary exploration, and may even deviate from an initially correct solution after prolonged deliberation (Chen et al., 2025b; Zhou et al., 2026; Zhang et al., 2026; Li et al., 2026). Conversely, overly restricting reasoning can hurt difficult problems that genuinely require more computation (Madaan et al., 2023; Jin et al., 2024). The key challenge is therefore not simply to make reasoning shorter or longer, but to determine whether additional reasoning remains beneficial and to allocate test-time computation accordingly.

This challenge has a close analogue to the classical concept of "metacognition" in cognitive science, which is commonly characterized as the monitoring and control of one's own cognitive processes (Ackerman & Thompson, 2017). The Nelson–Narens framework (Nelson, 1990) characterizes metacognition as the interaction between a meta-level and an object-level processes, with the former monitoring and steering the state of the latter. Applying this concept to reasoning LLMs, it is natural to propose the idea of *metareasoning* strategy that regulates cognitive effort and strategy, including whether to continue, redirect, or terminate a reasoning process (Lieder & Griffiths, 2017).

We posit that overthinking in LRMs can be understood as a failure of resource-rational metareasoning. At any point in a reasoning trajectory, additional computation is worthwhile only when its expected benefit outweighs its cost. Continuing after this marginal utility has vanished leads to overthinking, whereas terminating while further computation is still valuable leads to underthinking. Crucially, the computational demand of a problem is not determined by the input alone(Liu et al., 2026b; Jia et al., 2026): it depends on the reasoner's capability and evolves dynamically as the reasoning trajectory unfolds. For example, the same problem may be trivial for a stronger reasoner yet require substantial deliberation for a weaker one. Moreover, the same problem may require substantial computation early on, yet little or none once a promising solution path or sufficient solution has been reached. Therefore, efficient reasoning should be viewed as a dynamic resource-allocation problem, where the value of further reasoning must be continually reassessed as the process unfolds.

Existing approaches only partially address this dynamic allocation problem. Prompt-based methods (Aytes et al., 2025; Ding et al., 2024; Han et al., 2025; Xu et al., 2025) estimate problem difficulty from the input and predict a reasoning budget/strategy before generation. However, the input alone provides only limited evidence about how much computation a particular reasoner will need, making the accurate allocation difficult. Model-based methods (Liu et al., 2026b; Yang et al., 2025d; Ma et al., 2025b; Xia et al., 2025) use supervised fine-tuning (SFT) or reinforcement learning (RL) to induce concise or adaptive reasoning, but they entangle object-level reasoning with meta-level regulation, which may cause disturbance to the reasoner. Output-based methods (Li et al., 2026; Wang et al., 2025; Yang et al., 2026; Huang et al., 2026) instead regulate computation during inference through compression, pruning, early termination, or other interventions, but often rely on predefined proxy signals or decision rules whose reliability may vary across scenarios. While effective, these approaches do not fully address the trajectory-level challenge of regulating computation online, without retraining the reasoner or relying on predefined allocation rules.

This gap naturally motivates a resource-rational metacognitive approach to reasoning control, where computation is allocated online according to the problem and the evolving reasoning trajectory rather than prescribed in advance or encoded into the reasoner itself. We therefore formulate efficient reasoning as a sequential metacognitive control problem and introduce **MetaCtrl**, a learned meta-level controller that regulates a frozen object-level reasoner. The controller repeatedly observes the evolving reasoning state and dynamically determines how reasoning should proceed through four lightweight interventions: *continue* leaves reasoning unchanged, *fast-think* encourages concise derivation, *skip-think* bypasses redundant intermediate reasoning, and *stop-think* triggers a bounded conclusion before terminating deliberation, while preserving the reasoning already generated. Rather than relying on supervised intervention trajectories, fixed human-specified reasoning budgets, or hand-crafted rules, we train the controller directly with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to learn when and how to regulate reasoning from trajectory-level

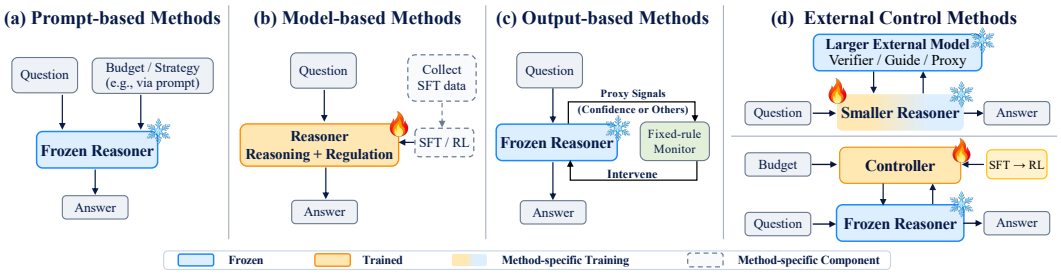


Figure 2: Comparison of four paradigms for efficient LRMs inference. In external control methods, some approaches train the reasoner, while others keep it frozen.

outcomes. The reward signal jointly reflects final correctness and reasoning efficiency, providing credit to the sequence of meta-level decisions according to their eventual effect on task success and computation cost.

We evaluate **MetaCtrl** on seven reasoning benchmarks spanning mathematics, science, and code. On the seen DeepSeek-R1-Distill-Qwen-7B reasoner, MetaCtrl improves accuracy while reducing generation length by **47.3%**, **58.2%**, and **72.1%** on mathematical, scientific, and code reasoning, respectively. More importantly, the controller transfers to multiple unseen reasoners without model-specific training. On Qwen3-14B, it reduces generation length by **51.6%**, **62.4%**, and **33.2%** across the three domains, respectively, while improving accuracy across all evaluated benchmarks. Notably, compared with DeepSeek-V4.1-Flash (552B parameters) (Xu et al., 2026) as the controller under the same frozen reasoner, MetaCtrl (4B parameters) improves average accuracy by 7.6 percentage points while reducing average generation length by 7.3% across the five benchmarks. Overall, our contributions can be outlined as follows:

(1) Resource-rational metacognitive formulation. We frame overthinking and underthinking in LRMs as failures to appropriately regulate inference-time computation, and introduce **MetaCtrl**, a resource-rational metacognitive framework that dynamically regulates reasoning effort according to the reasoner and its evolving reasoning trajectory.

(2) Learning metacognitive control from outcomes. To learn when and how to regulate reasoning, we introduce a metacognitive policy learning approach that uses GRPO to optimize the sequential interventions in a frozen reasoner. Final correctness and reasoning efficiency jointly guide policy learning, eliminating the need for supervised intervention trajectories.

(3) Effective and transferable reasoning regulation. Comprehensive experiments show that MetaCtrl consistently improves the reasoning accuracy and efficiency of LRMs over strong baselines, and generalizes well to unseen reasoning models without additional training.

2 Related Work

Following Sui et al. (2025), we group efficient reasoning methods into three categories and additionally discuss external reasoning control methods that are most closely related to ours, as illustrated in Figure 2. **Prompt-based methods** improve efficiency without modifying reasoner parameters (Lee et al., 2025; Yu et al., 2025a; Zhang et al., 2025; Renze & Guven, 2024; Xu et al., 2025; Han et al., 2025; Aytes et al., 2025; Liang et al., 2025; Liu et al., 2026a). They shorten reasoning or select budgets and reasoning strategies from the input. However, it is difficult to determine from the question alone how much and how to reason, and decisions made before generation cannot adapt to subsequent changes in reasoning progress. MetaCtrl instead regulates computation online by conditioning interventions on the evolving reasoning trajectory. **Model-based methods** internalize efficient reasoning through post-training (Kang et al., 2025; Ma et al., 2025b; Yang et al., 2025b; Yu et al., 2024; Xia et al., 2025; Luo et al., 2026; Hou et al., 2025; Aggarwal & Welleck, 2025; Shen et al., 2025; Liu et al., 2026b). These approaches train the reasoner itself to shorten, compress, or adapt its reasoning, often through SFT, RL, or explicit length constraints. In contrast, MetaCtrl freezes the reasoner and decouples problem solving from computation control, enabling the same controller to transfer across reasoners without retraining them. This separation becomes increas-

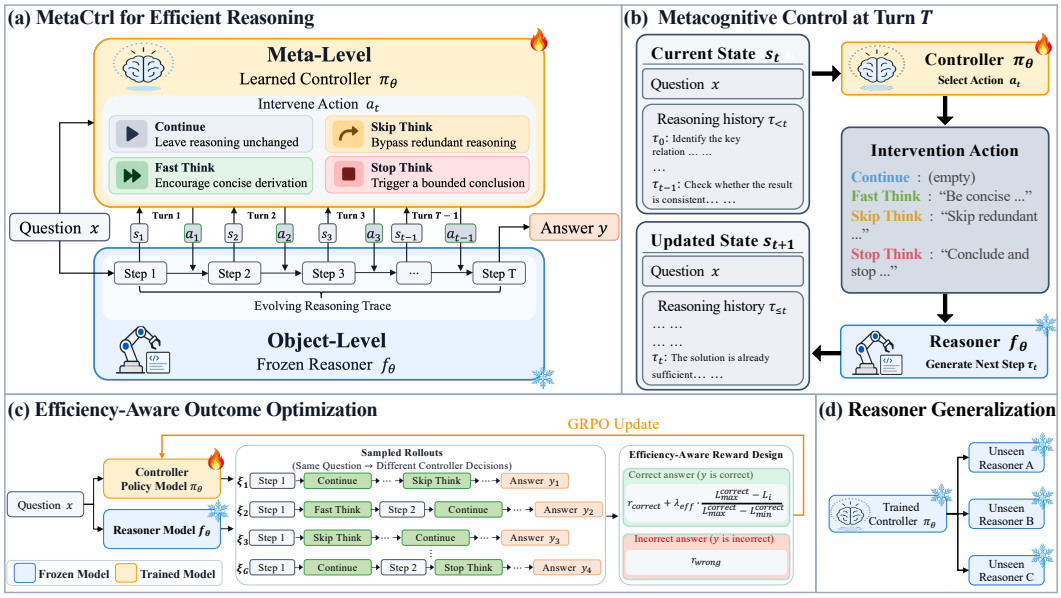


Figure 3: **Overview of MetaCtrl.** (a) A learned meta-level controller regulates a frozen reasoner through four intervention actions, without an input budget. (b) At each turn, it selects an intervention based on the current reasoning state. (c) The controller is trained with GRPO using an efficiency-aware outcome reward. (d) The trained controller generalizes to unseen reasoners without retraining.

ingly advantageous as reasoner scale grows, since only the lightweight controller requires training. **Output-based methods** regulate ongoing generation using signals such as confidence, certainty, hidden states, attention, or logits (Liao et al., 2025; Liu & Wang, 2025; Tikhonov et al., 2026; Hao et al., 2024; Yang et al., 2026; Fu et al., 2025; Chen et al., 2025a; Li et al., 2026). Many rely on predefined decision rules or thresholds, or require access to internal model signals, making their behavior sensitive to the chosen control criteria. MetaCtrl instead learns trajectory-conditioned interventions directly from correctness–efficiency rewards using only the observable reasoning trace. **External reasoning control** uses an auxiliary model to regulate a separate reasoner (Pan et al., 2025; Yang et al., 2025c; Wang et al., 2026; Xia et al., 2026). Prior methods may rely on substantially larger external models or even human feedback, and often participate directly in problem solving by generating, verifying, correcting, or providing solution-specific guidance, effectively coupling reasoning across multiple models. Others additionally retrain the reasoner or require distilled supervision and predefined token budgets. MetaCtrl instead learns a lightweight, budget-free controller through reinforcement learning, using only task-agnostic control actions to regulate a frozen reasoner without supplying solution content or requiring supervised intervention trajectories. Appendix A provides a more detailed discussion.

3 METACTRL: METACOGNITIVE REASONING CONTROL

3.1 GENERATION WITH METACOGNITIVE CONTROL

We formulate reasoning as a generation process controlled by metacognitive actions. Let x denote an input problem, f_ϕ a frozen object-level reasoner, and π_θ a trainable meta-level controller. We assume that the final successful reasoning trajectory can be decomposed into multiple sub-traces:

$$[\tau_0, a_1, \tau_1, \ldots, a_T, \tau_T],$$

in which a_t is an **action** sampled from the controller's discrete output space $\mathcal{A}$ (section 3.2), determined by all preceding traces and actions:

$$a_t \sim \pi_\theta(\cdot \mid x \oplus \tau_0 \oplus a_1 \cdots \oplus a_{t-1} \oplus \tau_{t-1}),$$

where $\oplus$ denotes concatenation of traces and actions with the relative order preserved. Let s_t denote the condition as a state, then the action selection essentially from a Markovian policy $\pi_\theta(\cdot \mid s_t)$.

The newly selected action a_t produces a new trace τ_t from the reasoner f_ϕ. This process repeats for T steps and generate a full controlled reasoning trajectory $\xi = (\tau_0, a_1, \tau_1, \ldots, a_T, \tau_T)$, which terminates with a final answer y_ξ. Therefore, given a dataset $\mathcal{D}$, we optimize the controller θ to balance reasoning performance and efficiency:

$$\max_{\theta} \ \mathbb{E}_{(x,y^*)\sim\mathcal{D},\, \xi\sim p_{\theta,\phi}(\cdot|x)} \left[R(\xi, y^*)\right],$$

where y^* denotes the ground-truth answer and R jointly evaluates answer correctness and object-level reasoning cost. The reasoner parameters ϕ remain frozen throughout training.

3.2 Action Space of Metacognitive Control

The action space of metacognitive controls a_t matter to the final performance. MetaCtrl's basic principle is to place a lightweight text-modality interventions on the evolving reasoning trace. Concretely, at each control point, we select from a discrete action space

$$\mathcal{A} = \{\textsc{Continue}, \textsc{Fast Think}, \textsc{Skip Think}, \textsc{Stop Think}\}.$$

CONTINUE preserves the context and lets the reasoner proceed naturally. FAST THINK encourages a more concise continuation, while SKIP THINK more aggressively directs the reasoner toward the solution with only the necessary intermediate reasoning. STOP THINK signals that the current reasoning is sufficient and prompts the reasoner to conclude and finalize its answer. Except for CONTINUE, each action is implemented by appending a short textual prompt to the current context. The exact prompts are provided in Appendix B.1.

Where to intervene. Control is exposed only at natural language boundaries. After the initial uncontrolled trace τ_0, we let generation pause at sentence-ending paragraph boundaries, implemented using `".\n\n"`, `"?\n\n"`, and two whitespace variants `". \n\n"`, `"? \n\n"`. The controller is invoked between coherent reasoning traces rather than at arbitrary token positions, providing semantically meaningful states while leaving token-level generation *uninterrupted*.

When to intervene. The action is sampled from the policy distribution $\pi_\theta(\cdot \mid s_t)$ conditioned on the evolving interaction. In contrast, problem-level allocation methods select a reasoning strategy or budget before generation according to $\pi(\cdot \mid x)$, while output-based methods often trigger early exit through predefined proxy criteria, e.g., stopping when the confidence C_t of an induced trial answer exceeds a fixed threshold δ. Conditioning on s_t allows MetaCtrl to determine when to intervene from the semantic content of the reasoning trace, enabling flexible control as reasoning evolves.

3.3 Outcome-Guided Policy Optimization for Efficient Metareasoning

We train MetaCtrl directly from the outcomes of its interactions with the frozen reasoner, without expert intervention trajectories or intermediate action labels. For each problem x, we sample a group of G controlled reasoning trajectories $\{\xi_i\}_{i=1}^{G}$ using the controller π_θ, and evaluate each trajectory by both answer correctness and reasoner generation length.

Correctness-gated efficiency reward. Let $c_i \in \{0, 1\}$ indicate whether ξ_i produces the correct answer, and let L_i be the total number of tokens generated by the object-level reasoner, including both reasoning and final answer generation.

To prevent efficiency optimization from favoring prematurely terminated but incorrect reasoning, we reward shorter computation *only among correct trajectories*. Define

$$\mathcal{I}^{\text{correct}} = \{i \mid c_i = 1\}, \qquad L_{\min}^{\text{correct}} = \min_{i\in\mathcal{I}^{\text{correct}}} L_i, \qquad L_{\max}^{\text{correct}} = \max_{i\in\mathcal{I}^{\text{correct}}} L_i.$$

For a correct trajectory, its relative efficiency is

$$e_i = \frac{L_{\max}^{\text{correct}} - L_i}{L_{\max}^{\text{correct}} - L_{\min}^{\text{correct}}},$$

with $e_i = 0$ when all correct trajectories have the same length. The reward is

$$R_i = \begin{cases} r_{\text{correct}} + \lambda_{\text{eff}} e_i, & c_i = 1, \\ r_{\text{wrong}}, & c_i = 0, \end{cases}$$

Table 1: Comparison across reasoning models of different scales and and multiple baselines.

Methods	Math Reasoning									Scientific Reasoning			AVG	
	MATH-500			AIME2024			OmniMath			GPQA Diamond				
	Acc.	Len.	Reduc.	Acc.	Len.	Reduc.	Acc.	Len.	Reduc.	Acc.	Len.	Reduc.	Acc.↑	Reduc.↑
***DeepSeek-R1-Distill-Qwen-7B* [Seen Reasoner]**														
NoThinking	79.4	699	-75.5	40.0	4134	-60.9	40.0	2083	-63.7	29.3	2084	-61.0	47.2	-65.3
Vanilla	85.2	2857	−0%	50.0	10570	−0%	45.0	5736	−0%	32.8	5349	−0%	53.3	−0%
D-Prompt	86.0	2726	−4.6%	46.7	10209	−3.4%	45.0	5398	−5.9%	30.3	5117	−4.3%	52.0	−4.6%
TokenSkip	80.2	2109	−26.2%	40.0	5559	−47.4%	38.3	3461	−39.7%	28.8	4001	−25.2%	46.8	−34.6%
CoT-Valve	78.6	**747**	−73.9%	43.3	**5871**	−44.5%	41.7	3311	−42.3%	28.3	3226	−39.7%	48.0	−50.1%
AdaCtrl	74.0	3196	+11.9%	21.3	16889	+59.8%	-	-	-	-	-	-	-	-
TH2T	86.8	1772	−38.0%	50.0	7490	−29.1%	48.3	3377	−41.1%	32.8	4166	−22.1%	54.5	−32.6%
ACTS	85.0	2181	−23.7%	36.7	5909	−44.1%	46.4	4622	−19.4%	40.9	5358	+0.2%	52.3	−21.8%
MetaCtrl	**88.0**	1644	−42.5%	**56.7**	5899	−44.2%	**48.7**	**3197**	−44.3%	**42.4**	**2234**	−58.2%	**59.0**	−47.3%
***DeepSeek-R1-Distill-Qwen-32B* [Unseen Reasoner]**														
NoThinking	80.2	659	-72.0	53.3	3208	-66.6	45.0	2146	-61.3	50.5	1627	-62.9	57.3	-65.7
Vanilla	87.2	2357	−0%	**60.0**	9605	−0%	50.0	5540	−0%	52.0	4384	−0%	**62.3**	−0%
D-Prompt	86.8	2198	−6.8%	56.7	9445	−1.7%	48.3	5135	−7.3%	49.5	3877	−11.6%	60.3	−6.9%
TokenSkip	79.8	1567	−33.5%	50.0	5856	−39.0%	43.3	3730	−32.7%	47.4	2874	−34.4%	55.1	−34.9%
CoT-Valve	85.3	2263	−3.9%	53.3	7997	−16.7%	43.3	4663	−15.8%	49.0	3550	−19.0%	57.7	−13.9%
TH2T	**88.0**	1733	−26.5%	56.7	7004	−27.1%	48.3	**2998**	−45.9%	52.0	2367	−46.0%	61.3	−36.4%
ACTS	85.0	2132	−9.5%	43.3	6323	−34.2%	50.3	4539	−18.1%	52.5	5036	+14.9%	57.8	−11.7%
MetaCtrl	**88.0**	**1528**	−35.2%	56.7	**4951**	−48.5%	**50.6**	3113	−43.8%	**53.2**	**1968**	−55.1%	62.1	−45.6%

where $r_{\text{correct}} > 0$ is the base correctness reward, $\lambda_{\text{eff}} > 0$ controls the strength of the efficiency bonus, and $r_{\text{wrong}} < 0$ is the penalty for an incorrect trajectory. Thus, shorter trajectories are favored only among correct solutions, while all incorrect trajectories receive the same length-independent penalty, encouraging efficient reasoning without trading correctness for premature termination.

Online GRPO update. We optimize the controller online with GRPO (Shao et al., 2024) over trajectories sampled for the same problem. Following the group-normalization treatment of Dr. GRPO (Liu et al., 2025), we omit within-group standard-deviation normalization and use the mean-centered advantage

$$\hat{A}_i = R_i - \bar{R}, \qquad \bar{R} = \frac{1}{G}\sum_{j=1}^{G} R_j.$$

The trajectory-level advantage $\hat{A}_i$ is assigned to the controller generated action tokens in ξ_i, while reasoner generated tokens are masked from the policy loss. Consequently, policy optimization updates only π_θ, with f_ϕ remaining fixed.

Through outcome-guided optimization, MetaCtrl learns trajectory-dependent interventions that improve both reasoning correctness and efficiency, without expert intervention trajectories or intermediate action labels. Rather than imitating a fixed reasoning schedule, the controller learns to allocate computation dynamically from the relative outcomes of its own interactions with the frozen reasoner.

4 Experiments

4.1 Experimental Setup

We evaluate our method on seven benchmarks spanning mathematical reasoning, scientific reasoning, and code reasoning, using multiple target reasoners and comparing against a broad set of recent and strong baselines. Full details of the training data, evaluation benchmarks and metrics, backbone models, baselines and implementation settings are provided in Appendix C, while prompting details are provided in Appendix B.

Table 2: Comparison across the Qwen3 series at different model scales and multiple baselines.

Methods	Math Reasoning									Scientific Reasoning			AVG	
	MATH-500			AIME 2024			OlympiadBench			GPQA Diamond				
	Acc.	Len.	Reduc.	Acc.	Len.	Reduc.	Acc.	Len.	Reduc.	Acc.	Len.	Reduc.	Acc.↑	Reduc.↑
***Qwen3-8B* [Unseen Reasoner]**														
NoThinking	87.4	1480	-70.0	23.3	7121	-41.2	48.7	5219	-43.7	48.5	2564	-72.7	52.0	-56.9
Vanilla	92.2	4926	−0%	63.3	12101	−0%	59.9	9268	−0%	52.5	9382	−0%	67.0	−0%
TALE	91.8	3682	−25.3%	60.0	11847	−2.1%	56.1	7306	−21.2%	52.0	4161	−55.6%	65.0	−26.0%
Dynasor	92.4	3361	−31.8%	60.0	11365	−6.1%	62.4	7640	−17.6%	55.1	5647	−39.8%	67.5	−23.8%
DEER	92.4	2966	−39.8%	63.3	8930	−26.2%	61.7	7367	−20.5%	54.5	5334	−43.1%	68.0	−32.4%
ASAG	93.0	3152	−36.0%	**66.7**	8683	−28.3%	63.9	7296	−21.3%	56.1	5714	−39.1%	69.9	−31.2%
ACTS	90.2	2736	−44.5%	46.7	**7148**	−40.9%	58.0	5059	−45.4%	**57.6**	5102	−45.6%	63.1	−44.1%
MetaCtrl	**94.0**	**2294**	−53.4%	**66.7**	8645	−28.6%	**64.0**	**4134**	−55.4%	57.1	**3518**	−62.5%	**70.5**	−50.0%
***Qwen3-14B* [Unseen Reasoner]**														
NoThinking	88.0	1403	-70.3	30.0	7726	-26.7	50.4	5462	-35.9	50.5	2308	-70.0	54.7	-50.7
Vanilla	93.4	4725	−0%	70.0	10537	−0%	61.9	8519	−0%	58.6	7688	−0%	71.0	−0%
TALE	93.2	3616	−23.5%	70.0	10358	−1.7%	58.4	7250	−14.9%	59.1	5391	−29.9%	70.2	−17.5%
Dynasor	93.8	3958	−16.2%	70.0	11156	+5.9%	63.0	7336	−13.9%	59.6	5387	−29.9%	71.6	−13.5%
DEER	94.0	3410	−27.8%	**73.3**	8402	−20.3%	62.7	7020	−17.6%	60.6	4998	−35.0%	72.7	−25.2%
ASAG	95.0	3505	−25.8%	**73.3**	7602	−27.9%	**64.6**	6819	−20.0%	63.1	5075	−34.0%	74.0	−26.9%
ACTS	92.6	2664	−43.6%	53.3	6861	−34.9%	60.4	4875	−42.8%	59.6	4859	−36.8%	66.5	−39.5%
MetaCtrl	**95.2**	**1938**	−59.0%	**73.3**	**6627**	−37.1%	64.4	**3712**	−56.4%	**63.6**	**2890**	−62.4%	**74.1**	−53.7%

Table 3: Baseline comparison with frontier LLMs as controllers. The reasoning model is DeepSeek-R1-Distill-Qwen-7B.

Controller Model	MATH-500			AMC			AIME2024			OlympiadBench			GPQA Diamond		
	Acc.	Len.	Reduc.	Acc.	Len.	Reduc.	Acc.	Len.	Reduc.	Acc.	Len.	Reduc.	Acc.	Len.	Reduc.
Vanilla (No Controller)	85.2	2857		73.5	7279		50.0	10570		48.6	8347		32.8	5349	
Qwen-3.8-Flash	71.4	6548	+129.2%	60.2	9229	+26.8%	–	–	–	–	–	–	30.3	10584	+97.9%
Qwen-3.8-Max	78.2	1605	-43.8%	48.8	2507	-65.6%	30.0	7780	-26.4%	39.3	2921	-65.0%	38.4	1813	-66.1%
GLM-5.2	82.0	1472	-48.5%	66.9	2450	-66.3%	36.7	5524	-47.7%	43.0	2355	-71.8%	40.4	1930	-63.9%
DeepSeek-V4.1-Flash	84.6	1781	-37.7%	68.7	2892	-60.3%	36.0	5949	-43.7%	48.7	3371	-59.6%	38.0	2822	-47.2%
GPT-5.6-Luna	76.4	**1326**	**-53.6%**	53.0	**1767**	**-75.7%**	30.0	**4320**	**-59.1%**	37.6	**1835**	**-78.0%**	38.9	**1665**	**-68.9%**
MetaCtrl	**88.0**	1644	-42.5%	**74.7**	3036	-58.3%	**56.7**	5899	-44.2%	**52.3**	2965	-64.5%	**42.4**	2234	-58.2%

4.2 Main Results

MetaCtrl improves reasoning accuracy while reducing redundant reasoning. As shown in Tables 1, on the seen DeepSeek-R1-Distill-Qwen-7B reasoner, MetaCtrl achieves the highest accuracy on all four benchmarks, improving average accuracy from 53.3% to 59.0% over Vanilla while reducing total generated tokens by 47.3%. Compared with D-Prompt, TokenSkip, TH2T, and ACTS, MetaCtrl improves average accuracy by 7.0, 12.2, 4.5, and 6.7 percentage points, while achieving an additional 42.7, 12.7, 14.7, and 25.5 points of token reduction, respectively. NoThinking and CoT-Valve exhibit more aggressive compression, with 18.0 and 2.8 points greater token reduction than MetaCtrl, but at substantial accuracy costs of 11.8 and 11.0 points. Notably, ACTS is the closest external control baseline, but requires a predefined token budget that directly affects both accuracy and generation length (Xia et al., 2026) and can be difficult to specify a priori, as the computation required for a correct solution varies across problems and reasoners. Overall, MetaCtrl achieves a consistently stronger accuracy–efficiency trade-off, suggesting that trajectory-dependent metacognitive control can reduce redundant reasoning while preserving or improving reasoning accuracy. More detailed comparisons across different baseline categories, end to end latency measurements, and case studies are provided in Appendices D, F, and I, respectively.

MetaCtrl generalizes effectively to unseen reasoners. Beyond substantially reducing redundant reasoning and improving accuracy on the training-time reasoner, MetaCtrl transfers directly to unseen reasoners without any further training, while retaining strong accuracy–efficiency trade-offs. As shown in Tables 2, on Qwen3-8B and Qwen3-14B, MetaCtrl improves average accuracy from

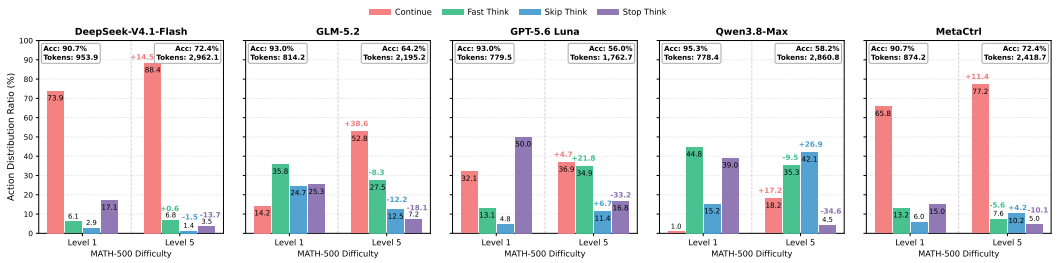


Figure 4: Action distributions of frontier LLMs as controllers and MetaCtrl across MATH-500 difficulty levels. The reasoning model is DeepSeek-R1-Distill-Qwen-7B.

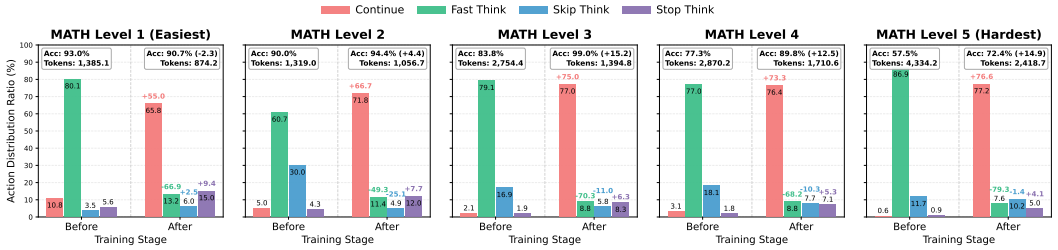


Figure 5: Action distribution change by MetaCtrl training. The reasoning model is DeepSeek-R1-Distill-Qwen-7B.

67.0% to 70.5% and from 71.0% to 74.1%, respectively, while reducing total generated tokens by 50.0% and 53.7%. Compared with ASAG, the strongest baseline on these reasoners, MetaCtrl achieves comparable or higher accuracy while substantially larger token reductions (50.0% vs. 31.2% on Qwen3-8B and 53.7% vs. 26.9% on Qwen3-14B). As shown in Tables 1, on DeepSeek-R1-Distill-Qwen-32B, MetaCtrl maintains accuracy close to Vanilla, achieving 62.1% versus 62.3%, while reducing generation length by 45.6%. These results indicate that MetaCtrl learns transferable metacognitive control that generalizes across different reasoners and model scales without additional post training. Generalization to additional reasoner families is provided in Appendix E.1.

MetaCtrl generalizes across benchmarks and domains. As shown in Tables 1 and 2, MetaCtrl consistently achieves strong accuracy with substantially shorter generation lengths across mathematical and scientific reasoning tasks. On MATH-500, it improves or maintains high accuracy across all four reasoners while reducing generation length by 35.2% to 59.0%. Similar gains are observed on more challenging mathematical benchmarks, including AIME 2024, OmniMath, and OlympiadBench, where MetaCtrl substantially shortens reasoning trajectories while maintaining competitive accuracy. The same trend extends to GPQA Diamond, where MetaCtrl achieves accuracies of 42.4%, 53.2%, 57.1%, and 63.6% across the four reasoners, with generation reductions ranging from 55.1% to 62.5%. These results suggest that MetaCtrl provides effective reasoning control across diverse domains and problem difficulty levels. Additional code reasoning results are reported in Appendix E.1.

Comparison with frontier LLM controllers. Table 3 shows that stronger general purpose LLMs do not necessarily provide more effective reasoning control. MetaCtrl achieves the highest accuracy on all five benchmarks, outperforming the best frontier LLM controller by 2.0 to 20.0 percentage points. Although some frontier LLM controllers achieve larger token reductions on individual benchmarks, these gains often coincide with substantial accuracy losses. As shown in Figure 4, GLM-5.2, GPT-5.6-Luna, and Qwen-3.8-Max frequently apply aggressive shortening actions to difficult MATH 500 problems, coinciding with marked accuracy drops, whereas MetaCtrl better preserves continued reasoning on difficult problems while still reducing computation. These results suggest that effective control requires selectively reducing unnecessary computation while retaining the reasoning needed for correctness, rather than simply shortening trajectories.

Adaptive Reasoning Allocation by Difficulty. Figure 4 examines how control policies vary with problem difficulty on MATH-500. For MetaCtrl, *Continue* increases from 65.8% at Level 1 to 77.2% at Level 5, while *Stop Think* decreases from 15.0% to 5.0%, suggesting that the controller

Table 4: Ablation study on reward-function coefficients. The reasoning model is DeepSeek-R1-Distill-Qwen-7B.

Reward Setting			MATH-500			AMC			AIME2024			OlympiadBench			GPQA Diamond		
r_{correct}	r_{wrong}	λ_{eff}	**Acc.**	**Len.**	**Reduc.**	**Acc.**	**Len.**	**Reduc.**	**Acc.**	**Len.**	**Reduc.**	**Acc.**	**Len.**	**Reduc.**	**Acc.**	**Len.**	**Reduc.**
Vanilla (No Controller)			85.2	2857		73.5	7279		50.0	10570		48.6	8347		32.8	5349	
1.5	-1.0	0.5	88.8	1739	-39.1%	76.1	3175	-56.4%	40.7	6261	-40.8%	52.3	3092	-63.0%	**43.4**	2789	-47.9%
2.0	-1.0	0.5	**90.8**	1921	-32.8%	**77.6**	4162	-42.8%	44.0	8106	-23.3%	**53.3**	3870	-53.6%	37.9	4818	-9.9%
1.0	-0.5	0.5	85.8	1553	-45.6%	71.1	2545	-65.0%	40.7	5013	-52.6%	45.9	2590	-69.0%	36.9	1998	-62.6%
1.0	-1.0	1.0	82.0	**1284**	-55.1%	59.0	**2053**	-71.8%	30.7	**4626**	-56.2%	40.9	**1968**	-76.4%	37.5	**1463**	-72.6%
1.0	**-1.0**	**0.5**	88.0	1644	-42.5%	74.7	3036	-58.3%	**56.7**	5899	-44.2%	52.3	2965	-64.5%	42.4	2234	-58.2%

becomes increasingly likely to preserve ongoing reasoning as problem difficulty increases. A similar coarse trend appears across the frontier LLM controllers, all of which increase *Continue* and reduce *Stop Think* as difficulty rises. Among them, DeepSeek-V4.1-Flash exhibits the most similar action distributions to MetaCtrl across difficulty levels and also achieves the strongest accuracy among the frontier LLM controllers, matching MetaCtrl at both difficulty levels but with longer generations. Beyond this shared trend, the controllers differ in their finer grained adjustments, with GPT-5.6-Luna substantially increasing *Fast Think*, and Qwen3.8-Max increasing *Skip Think*. Overall, this difficulty-aware control suggests that MetaCtrl intervenes more conservatively on harder problems while applying stronger reasoning compression on easier ones, contributing to its strong accuracy with shorter generations across difficulty levels. Further analyses of control behavior across tasks and reasoners are provided in Appendix E.2.

MetaCtrl training learns effective reasoning control. Figure 5 compares the controller's action distributions before and after training across MATH-500 difficulty levels. Before training, the controller is dominated by *Fast Think*, whereas training shifts the policy strongly toward *Continue* at every difficulty level. On the hardest Level 5 problems, for example, *Continue* increases from 0.6% to 77.2%, while *Fast Think* decreases from 86.9% to 7.6%. This policy shift improves accuracy while reducing generation length on Levels 2 through 5, with accuracy gains of 12.5 to 15.2 percentage points and generation reductions of 40.4% to 49.4% on the harder Levels 3 through 5. Overall, these results suggest that effective reasoning control is not achieved by simply encouraging shorter or faster reasoning, but by learning when to preserve computation and when to reduce it. Further analyses of training dynamics and policy evolution are provided in Appendix G.

4.3 Ablation Study

Reward coefficients. Table 4 examines the effect of reward coefficients on accuracy and generation length. The default setting, $r_{\text{correct}} = 1.0$, $r_{\text{wrong}} = -1.0$, and $\lambda_{\text{eff}} = 0.5$, is the only tested configuration that improves accuracy over Vanilla on all five benchmarks, while reducing generation length by 42.5% to 64.5%. Weakening the wrong answer penalty to -0.5 yields stronger compression but lowers accuracy on several tasks, while increasing λ_{eff} to 1.0 further shortens generation at a larger accuracy cost. Increasing r_{correct} benefits some tasks but is less consistent across benchmarks. Overall, these results highlight the importance of balancing correctness preservation and reasoning compression. Additional ablation results are provided in Appendix E.3.

5 Conclusion

In this paper, we present MetaCtrl, a lightweight metacognitive controller that dynamically regulates a frozen reasoner based on its evolving reasoning trajectory. MetaCtrl learns when and how to intervene through GRPO with a correctness-gated efficiency objective, without supervised intervention trajectories, predefined reasoning budgets, or reasoner retraining. Extensive experiments across seven benchmarks show that MetaCtrl consistently improves the reasoning accuracy of controlled LRMs while substantially reducing their reasoning length, and generalizes effectively to unseen LRMs spanning different model scales and architectures as well as across diverse reasoning domains. With matched GPU memory, MetaCtrl also substantially reduces end to end LRM inference latency. These results demonstrate that decoupling metacognitive control from problem solving provides an effective and transferable mechanism for adaptively allocating reasoning computation.

AI USE STATEMENT

Generative AI tools were used solely to assist with language editing and writing refinement, including improving grammar, clarity, readability, and presentation of the manuscript. They were not used to develop the research methodology, design experiments, generate or analyze experimental results, or formulate the scientific conclusions of this work. All AI-assisted text was carefully reviewed, revised, and verified by the authors to ensure that it accurately reflects the intended technical content. The authors take full responsibility for the final content of the paper, including all statements, claims, and conclusions.

REFERENCES

Rakefet Ackerman and Valerie A Thompson. Meta-reasoning: Monitoring and control of thinking and reasoning. *Trends in cognitive sciences*, 21(8):607–617, 2017.

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

AI-MO. AMC 2023, 2024. URL https://huggingface.co/datasets/AI-MO/aimo-validation-amc.

Simon A Aytes, Jinheon Baek, and Sung Ju Hwang. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 24307–24331, 2025.

Lang Cao, Yingtian Zou, Chao Peng, Renhong Chen, Wu Ning, and Yitong Li. Step guided reasoning: Improving mathematical reasoning using guidance generation and step reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21112–21129, 2025.

Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. Seal: Steerable reasoning calibration of large language models for free. *arXiv preprint arXiv:2504.07986*, 2025a.

Xin Chen, Feng Jiang, Yiqian Zhang, Hardy Chen, Shuo Yan, Wenya Xie, Min Yang, and Shujian Huang. Reasoning while asking: Transforming reasoning large language models from passive solvers to proactive inquirers. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 35069–35090, 2026.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of long reasoning models. In *Forty-second International Conference on Machine Learning*, 2025b.

Mengru Ding, Hanmeng Liu, Zhizhang Fu, Jian Song, Wenbo Xie, and Yue Zhang. Break the chain: Large language models can be shortcut reasoners. *arXiv preprint arXiv:2406.06580*, 2024.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and Hao Zhang. Efficiently scaling LLM reasoning programs with certaindex. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=nn51ewu5k2.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Zhengyang Tang, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 100540–100569, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, 2025.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

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.

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

Jiameng Huang, Baijiong Lin, Guhao Feng, Jierun Chen, Di He, and Lu Hou. Efficient reasoning for large reasoning language models via certainty-guided reflection suppression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31176–31184, 2026.

Shijue Huang, Hongru Wang, Wanjun Zhong, Zhaochen Su, Jiazhan Feng, Bowen Cao, and Yi R Fung. Adactrl: Towards adaptive and controllable reasoning via difficulty-aware budgeting. *arXiv preprint arXiv:2505.18822*, 2025.

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL `https://github.com/huggingface/open-r1`.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Yaning Jia, Chunhui Zhang, Xingjian Diao, Xiangchi Yuan, Zhongyu Ouyang, Chiyu Ma, and Soroush Vosoughi. What makes a good curriculum? disentangling the effects of data ordering on llm mathematical reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34472–34488, 2026.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. The impact of reasoning step length on large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1830–1842, 2024.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24312–24320, 2025.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

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

Dacheng Li, Shiyi Cao, Chengkun Cao, Xiuyu Li, Shangyin Tan, Kurt Keutzer, Jiarong Xing, Joseph E Gonzalez, and Ion Stoica. S*: Test time scaling for code generation. In *EMNLP (Findings)*, pp. 15964–15978, 2025.

Jiakai Li, Ke Qin, Rongzheng Wang, Yizhuo Ma, Qizhi Chen, Muquan Li, and Shuang Liang. Stop when further reasoning won't help: Attention-state adaptive generation in reasoning models. *arXiv preprint arXiv:2606.15070*, 2026.

Guosheng Liang, Longguang Zhong, Ziyi Yang, and Xiaojun Quan. Thinkswitcher: When to think hard, when to think fast. In *EMNLP (Findings)*, pp. 5185–5201, 2025.

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*, 2025.

Falk Lieder and Thomas L Griffiths. Strategy selection as rational metareasoning. *Psychological review*, 124(6):762, 2017.

Xiang Liu, Xuming Hu, Xiaowen Chu, and Eunsol Choi. Diffadapt: Difficulty-adaptive reasoning for token-efficient llm inference. In *International Conference on Learning Representations*, volume 2026, pp. 31640–31666, 2026a.

Xin Liu and Lu Wang. Answer convergence as a signal for early stopping in reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 17907–17918, 2025.

Yongjiang Liu, Haoxi Li, Xiaosong Ma, Jie Zhang, and Song Guo. Think how to think: Mitigating overthinking with autonomous difficulty cognition in large reasoning models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 38105–38126, 2026b.

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

Haotian Luo, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Xiaochun Cao, Dacheng Tao, Naiqiang Tan, and Li Shen. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14242–14257, 2026.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*, 2025a.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. Cot-valve: Length-compressible chain-of-thought tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6025–6035, 2025b.

MAA Committees. AIME problems and solutions. URL `https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions`.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Thomas O Nelson. Metamemory: A theoretical framework and new findings. In *Psychology of learning and motivation*, volume 26, pp. 125–173. Elsevier, 1990.

Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. Specreason: Fast and accurate inference-time compute via speculative reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=wCbOKbZ7kf`.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

Qwen Team. Qwen3.8-max: A new bar for coding and cowork, August 2026. URL `https://qwen.ai/blog?id=qwen3.8`.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Matthew Renze and Erhan Guven. The benefits of a concise chain of thought on problem-solving in large language models. *arXiv preprint arXiv:2401.05618*, 2024.

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.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 2322–2331, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025.

Zayne Sprague, Fangcong Yin, Juan Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 94118–94162, 2025.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

Pavel Tikhonov, Ivan Oseledets, and Elena Tutubalina. Confidence leaps in llm reasoning: Early stopping and cross-model transfer. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 602–616, 2026.

Qianyue Wang, Jinwu Hu, Yaofo Chen, Yufeng Wang, Bailin Chen, Huanxiang Lin, Yu Rong, Yuanqing Li, Zhiquan Wen, and Mingkui Tan. Intervene when it doubts: Conjunction-guided interactive reasoning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=zVjLO7jg9T.

Yiming Wang, Pei Zhang, Siyuan Huang, Baosong Yang, Zhuosheng Zhang, Fei Huang, and Rui Wang. Sampling-efficient test-time scaling: Self-estimating the best-of-n sampling in early decoding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=BcKYVmh3yH.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3351–3363, 2025.

Yu Xia, Zhouhang Xie, Xin Xu, Byungkyu Kang, Prarit Lamba, Xiang Gao, and Julian McAuley. Agentic chain-of-thought steering for efficient and controllable llm reasoning. *arXiv preprint arXiv:2606.03965*, 2026.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

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

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weipinng Wang. Dynamic early exit in reasoning models. In *International Conference on Learning Representations*, volume 2026, pp. 88170–88210, 2026.

Junjie Yang, Ke Lin, and Xing Yu. Think when you need: Self-adaptive chain-of-thought learning. *arXiv preprint arXiv:2504.03234*, 2025b.

Wang Yang, Xiang Yue, Vipin Chaudhary, and Xiaotian Han. Speculative thinking: Enhancing small-model reasoning with large model guidance at inference time. *arXiv preprint arXiv:2504.12329*, 2025c.

Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. Towards thinking-optimal scaling of test-time compute for LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025d. URL https://openreview.net/forum?id=6ICFqmixlS.

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.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*, 2024.

Ye Yu, Yaoning Yu, and Haohan Wang. Premise: Scalable and strategic prompt optimization for efficient mathematical reasoning in large models. *arXiv preprint arXiv:2506.10716*, 2025a.

Yiyao Yu, Yuxiang Zhang, Dongdong Zhang, Xiao Liang, Hengyuan Zhang, Xingxing Zhang, Mahmoud Khademi, Hany Hassan Awadalla, Junjie Wang, Yujiu Yang, et al. Chain-of-reasoning: Towards unified mathematical reasoning in large language models via a multi-paradigm perspective. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24914–24937, 2025b.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Ruiqi Zhang, Changyi Xiao, and Yixin Cao. Long or short cot? investigating instance-level switch of large reasoning models. *arXiv preprint arXiv:2506.04182*, 2025.

Xinliang Frederick Zhang, Anhad Mohananey, Alexandra Chronopoulou, Pinelopi Papalampidi, Somit Gupta, Tsendsuren Munkhdalai, Lu Wang, and Shyam Upadhyay. Do llms really need 10+ thoughts for "find the time 1000 days later"? towards structural understanding of llm overthinking. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17005–17030, 2026.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

Shu Zhou, Rui Ling, Junan Chen, Xin Wang, Tao Fan, and Hao Wang. When more thinking hurts: Overthinking in llm test-time compute scaling. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 23967–23977, 2026.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An llm post-training framework for rl scaling. https://github.com/THUDM/slime, 2025. GitHub repository. Corresponding author: Xin Lv.

TABLE OF CONTENTS

A Related Work Details **16**

B Prompt Templates **17**

B.1 Intervention Prompts 17

B.2 Controller Prompt 17

C Experimental Details **18**

C.1 Datasets and Metrics. 18

C.2 Backbone Models. 18

C.3 Baseline Details 18

C.4 Implementation Details 19

D Detailed Comparison with Efficient Reasoning Baselines **19**

E Additional Experimental Results and Analysis **21**

E.1 Generalization Across Benchmarks, Domains, and Reasoning Model Architectures. 21

E.2 Adaptive Control Emerges from Task–Reasoner Interactions. 22

E.3 More Ablation Studies 23

F The Analysis of Latency and Computation Cost of Methods **24**

G Training Dynamics and Efficiency **24**

H Future work **25**

I Case Study **26**

I.1 Preventing Post-Solution Overthinking 26

I.2 Reducing Errors from Prolonged Reasoning 26

A Related Work Details

We provide a more detailed discussion of the efficient-reasoning paradigms summarized in Figure 2.

Prompt-based methods. Prompt-based approaches improve reasoning efficiency without modifying model parameters (Lee et al., 2025; Yu et al., 2025a; Zhang et al., 2025). Concise Chain-of-Thought (Renze & Guven, 2024) and Chain of Draft (Xu et al., 2025) prompt reasoning models to produce shorter and more concise reasoning traces, while TALE (Han et al., 2025) estimates question-specific token budgets before generation. Sketch-of-Thought (Aytes et al., 2025), ThinkSwitcher (Liang et al., 2025), and DiffAdapt (Liu et al., 2026a) further adapt reasoning formats or modes according to input characteristics. These methods can adapt computation to the input, but their main control decisions are made before or at the beginning of generation, when it is difficult to determine from the question alone how much computation is needed or how reasoning should proceed. Moreover, such decisions cannot respond to later changes in reasoning progress as the solution develops. MetaCtrl instead conditions its interventions on the evolving reasoning trace and adapts computation online throughout generation.

Model-based methods. Model-based approaches internalize efficient reasoning through post-training (Kang et al., 2025; Ma et al., 2025b; Yang et al., 2025b). System-2 distillation (Yu et al., 2024) and TokenSkip (Xia et al., 2025) compress deliberative reasoning, while O1-Pruner (Luo et al., 2026), ThinkPrune (Hou et al., 2025), and L1 (Aggarwal & Welleck, 2025) optimize reasoning length through reinforcement learning or explicit token constraints. DAST (Shen et al., 2025) and TH2T (Liu et al., 2026b) further incorporate difficulty or redundancy awareness to adapt computation across problems. These methods modify the reasoner itself and therefore require model-specific post-training, coupling problem solving with computation control. This is particularly costly for difficulty-aware methods because perceived problem difficulty is relative to the capability of the underlying reasoner, potentially requiring adaptation when the reasoner changes. MetaCtrl instead freezes the reasoner and trains only a lightweight meta-level controller, decoupling reasoning control from problem solving and avoiding costly reasoner training. This advantage becomes more pronounced as reasoner scale grows, while also enabling transfer across different reasoners without retraining them.

Output-based methods. Output-based approaches regulate the ongoing generation process or intervene on its internal representations (Liao et al., 2025; Liu & Wang, 2025; Tikhonov et al., 2026). Coconut (Hao et al., 2024) performs reasoning in continuous latent states rather than discrete language tokens. DEER (Yang et al., 2026) and Certaindex (Fu et al., 2025) use confidence or answer-stability signals to determine when additional computation is unnecessary, while SEAL (Chen et al., 2025a) intervenes through internal representations. ASAG (Li et al., 2026) combines model confidence and attention entropy to adapt generation strategies. These approaches often rely on predefined decision rules or thresholds, or require access to internal states such as hidden representations, attention, or logits. Moreover, fixed confidence-based criteria can behave differently across reasoning states and problem difficulty. MetaCtrl instead learns when and how to intervene directly from correctness–efficiency rewards using only the observable reasoning trajectory, without hand-crafted control criteria or access to internal model states.

External reasoning control. External control methods regulate a reasoner using a separate model. SpecReason (Pan et al., 2025) and Speculative Thinking (Yang et al., 2025c) use stronger models to verify, correct, or guide weaker reasoners. Although this can reduce computation performed by the target reasoner, the external model participates directly in problem solving by providing solution-specific reasoning guidance, effectively coupling inference across multiple capable agents and increasing system complexity. CGI (Wang et al., 2026) injects external state-based feedback from humans or LLM proxies. Its evaluator assesses the rationality and completeness of the current reasoning and can provide task-specific corrective guidance, thereby participating in the problem solving process to some extent. When an LLM proxy is used, this effectively forms a multi-model problem-solving system in which a stronger external model guides the target reasoner. CGI additionally trains the reasoner to respond to such interventions and may rely on an LLM proxy substantially larger than the target reasoner. ACTS (Xia et al., 2026) is more closely related to our setting because it controls a frozen reasoner with a separate controller. However, ACTS relies on distilled SFT data

and requires a predefined token budget, which can be difficult to determine a priori because the required computation depends jointly on problem difficulty and reasoner capability.

MetaCtrl differs from these approaches in three respects. First, it keeps the target reasoner fully frozen and requires no model-specific adaptation. Second, its controller does not participate in solving the task or provide solution-specific content. Instead, it selects a small set of task-agnostic control actions that determine how the reasoner should proceed. The controller is learned directly through reinforcement learning from evolving reasoning trajectories, without supervised intervention trajectories or distilled SFT data. Third, MetaCtrl requires neither a predefined reasoning budget nor a large external proxy, allowing a lightweight controller to dynamically regulate when and how reasoning proceeds as the solution unfolds.

B Prompt Templates

B.1 Intervention Prompts

MetaCtrl realizes its meta-level actions through lightweight textual prompts inserted into the reasoner's context at each control point. Table 5 reports the exact prompts used in our experiments. CONTINUE introduces no additional text, leaving the reasoning context unchanged. For the other actions, the corresponding prompt is appended to the current context before the frozen reasoner resumes generation. The leading newline used in implementation is omitted from the table for readability.

Table 5: Textual intervention prompts associated with the MetaCtrl action space.

Action	**Intervention Prompt**
CONTINUE	No textual intervention.
FAST THINK	`Let's keep only the essential derivation, avoid repetition, and proceed efficiently.`
SKIP THINK	`Let's solve this directly with the minimum necessary reasoning and move to the answer.`
STOP THINK	`The reasoning is sufficient; state only the final conclusion briefly and close the reasoning now.`

Importantly, STOP THINK does not forcibly truncate generation. Instead, it instructs the frozen reasoner to conclude its current reasoning and produce the final answer, preserving the same text-based interaction interface used by the other interventions.

B.2 Controller Prompt

The controller receives the original problem and the reasoner's latest reasoning trajectory. Its system prompt is:

> You control a frozen mathematical reasoner while it solves one problem.
>
> The reasoner always generates its first reasoning step without controller intervention. You are called only if reasoning is still active after that step. On every controller turn, you receive the reasoner's latest step and select one action from `continue`, `fast_think`, `skip_think`, or `stop_think`. Optimize for a correct answer first; among correct solutions, prefer the shortest sufficient reasoning.
>
> **Actions:**
>
> - `continue`: add no intervention and let the reasoner continue naturally.
> - `fast_think`: ask for only the essential derivation and no repetition.
> - `skip_think`: skip redundant intermediate work and move to the shortest remaining derivation.
> - `stop_think`: run one bounded conclusion continuation, close the thinking channel, and proceed to the final answer. Existing reasoning is preserved and never truncated.
>
> Output exactly one line and no explanation:
>
> `Action: <continue|fast_think|skip_think|stop_think>`

C Experimental Details

C.1 Datasets and Metrics.

We use 7,500 mathematical problems sampled from OpenR1-Math (Hugging Face, 2025; Xia et al., 2026) for online controller training. We evaluate across three reasoning domains: mathematics on MATH-500 (Hendrycks et al., 2021), AIME 2024 (MAA Committees), Omni-MATH (Gao et al., 2025), OlympiadBench (He et al., 2024), and AMC (AI-MO, 2024), science on GPQA Diamond (Rein et al., 2024), and code on LiveCodeBench (Jain et al., 2025). We report accuracy (pass@1), average token length, and length reduction ratio, which denotes the percentage reduction in token length relative to Vanilla. For controlled methods, token length includes generations from both the target reasoner and the controller.

C.2 Backbone Models.

During training, we initialize the controller from Qwen3-4B-Instruct-2507 (Yang et al., 2025a) and pair it with a frozen DeepSeek-R1-Distill-Qwen-7B reasoner (Guo et al., 2025). At evaluation, we test the controller on the training-time reasoner and assess cross-reasoner generalization by directly transferring it, without further training, to three unseen reasoners: DeepSeek-R1-Distill-Qwen-32B, DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025), Qwen3-8B, and Qwen3-14B (Yang et al., 2025a).

C.3 Baseline Details

We compare against a broad set of recent and strong baselines across six categories: (i) **Vanilla**: standard reasoning without intervention; (ii) **Prompt-based methods**: D-Prompt (Liu et al., 2026b), NoThinking (Ma et al., 2025a), and TALE (Han et al., 2025); (iii) **Model-based methods**: TokenSkip (Xia et al., 2025), CoT-Valve (Ma et al., 2025b), AdaCtrl (Huang et al., 2025) and TH2T (Liu et al., 2026b); (iv) **Output-based methods**: Dynasor (Fu et al., 2025), DEER (Yang et al., 2026), and ASAG (Li et al., 2026); and (v) **External reasoning control**: ACTS (Xia et al., 2026); and (vi) **Frontier LLM as controller**: GLM-5.2 (Zeng et al., 2026), Qwen3.8-Flash, Qwen3.8-Max (Qwen Team, 2026), DeepSeek-V4.1-Flash (Xu et al., 2026), and GPT-5.6-Luna[1],, each accessed through its official API and directly used as the controller under the same controller–reasoner interface.

Prompt-based methods. **D-Prompt** (Liu et al., 2026b) augments the input with a difficulty reminder and encourages the reasoner to adjust its response length according to its self-assessed problem difficulty. **NoThinking** (Ma et al., 2025a) instructs the model to bypass explicit intermediate reasoning and directly generate the answer. **TALE** (Han et al., 2025) controls reasoning length through token-budget prompting, assigning a constrained reasoning budget before generation.

Model-based methods. **TokenSkip** (Xia et al., 2025) fine-tunes the target reasoner to skip non-essential reasoning tokens during generation. **CoT-Valve** (Ma et al., 2025b) learns controllable directions in model parameter space to regulate chain-of-thought length. **TH2T** (Liu et al., 2026b) applies two-stage fine-tuning to induce difficulty and redundancy awareness, enabling adaptive reasoning depth while suppressing redundant reasoning. **AdaCtrl** (Huang et al., 2025) combines difficulty-aware fine-tuning with reinforcement learning to self-assess problem difficulty and adaptively allocate reasoning budgets.

Output-based methods. **Dynasor** (Fu et al., 2025) checks partial reasoning at regular intervals and stops generation once successive intermediate answers reach sufficient agreement. **DEER** (Yang et al., 2026) instead evaluates intermediate predictions at adaptive transition points in the reasoning process and terminates early when a high-confidence answer emerges. **ASAG** (Li et al., 2026) jointly uses model confidence and attention-state information to determine whether further reasoning is likely to be useful and adaptively controls generation.

[1] `https://developers.openai.com/api/docs/models/gpt-5.6-luna`

External reasoning control. **ACTS** (Xia et al., 2026) employs a separately trained controller to steer a frozen reasoner step by step. Its controller conditions on the evolving reasoning trajectory and an explicit remaining token budget, and is first initialized through supervised fine-tuning on steering data constructed from expert reasoning trajectories, followed by budget-conditioned reinforcement learning.

For the baseline methods, we directly report the results from the original TH2T (Liu et al., 2026b) and ASAG (Li et al., 2026) papers. For all evaluations of our method, we adopt the same configurations used in these prior works, ensuring fair comparison under matched inference settings. For ACTS (Xia et al., 2026), we reproduce its evaluation using greedy decoding for the reasoner to ensure a consistent decoding protocol across baselines. Implementation details are described in the Appendix C.4.

C.4 Implementation Details

Training and Serving. We directly train the controller with GRPO, without supervised fine-tuning. We use a learning rate of 1×10^{-6}, a GRPO group size of 8, a rollout batch size of 32, and a global batch size of 64. We set $r_{\text{correct}} = 1$, $r_{\text{wrong}} = -1$, and the efficiency coefficient $\lambda_{\text{eff}} = 0.5$. During training, the controller is sampled with temperature 1.0 and top-p 0.95, whereas the frozen reasoner uses greedy decoding. All experiments run on 8 NVIDIA H20 GPUs. RL training is implemented with SLIME (Zhu et al., 2025). At inference time, the controller and reasoner are hosted as separate SGLang (Zheng et al., 2024) servers.

Evaluation Protocol. For fair comparison, we directly report the baseline results from the original TH2T (Liu et al., 2026b) and ASAG (Li et al., 2026) papers, and evaluate our method using the same configurations adopted in these works across all experiments. Since TH2T and ASAG report results with greedy reasoner decoding, we also reproduce ACTS under greedy reasoner decoding rather than directly using its official evaluation setting, which samples from the reasoner with temperature 0.6 and top-p 0.95. Since ACTS requires an explicit input reasoning budget, we set the token budget to 2,000 for MATH-500, 5,000 for AIME 2024, and 4,000 for Omni-MATH, OlympiadBench, and GPQA Diamond. For our method, the controller follows the ACTS decoding configuration with temperature 0.7 and top-p 0.8, while the reasoner uniformly uses greedy decoding across all evaluations. The reasoner-side generation limits are matched to those of the corresponding baseline settings to avoid confounding performance differences with inference configuration.

D Detailed Comparison with Efficient Reasoning Baselines

Our main results are presented in Table 1 and Table 2. We provide a more detailed comparison with the four major families of efficient reasoning methods considered in our experiments. We focus on both empirical performance and how each paradigm allocates test-time computation.

Prompt-based methods. Prompt-based approaches regulate reasoning through instructions or token budgets specified before generation, and therefore cannot adapt their decisions to how the reasoning trajectory unfolds. This limitation is reflected in our results. On DeepSeek-R1-Distill-Qwen-7B, D-Prompt achieves 52.0% average accuracy with only a 4.6% token reduction, whereas MetaCtrl reaches 59.0% accuracy while reducing generation by 47.3%. NoThinking obtains a larger reduction of 65.3%, but its average accuracy drops to 47.2%, illustrating the cost of uniformly suppressing deliberation. The same trend holds for other reasoning models: compared with TALE, MetaCtrl improves average accuracy from 65.0% to 70.5% on Qwen3-8B and from 70.2% to 74.1% on Qwen3-14B, while increasing token reduction from 26.0% to 50.0% and from 17.5% to 53.7%, respectively. These results suggest that the amount of useful computation is difficult to determine from the input alone and is better adjusted according to the evolving reasoning state.

Model-based methods. Model-based methods improve reasoning efficiency by modifying the target reasoner itself through supervised fine-tuning or reinforcement learning. Such approaches can learn adaptive reasoning behaviors, but couple computation control with problem solving and generally require model-specific post-training. In contrast, MetaCtrl keeps the reasoner frozen and learns a separate metacognitive policy. On the DeepSeek-R1-Distill-Qwen-7B reasoner, TH2T, the strongest

model-based baseline in average accuracy, achieves 54.5% accuracy with a 32.6% token reduction, while MetaCtrl reaches 59.0% accuracy and a 47.3% reduction. TokenSkip and CoT-Valve achieve 46.8% and 48.0% average accuracy, respectively, despite reducing generation by 34.6% and 50.1%.

A further limitation of difficulty-aware model-based approaches such as TH2T is that perceived problem difficulty is inherently reasoner-dependent. Models at different scales can have substantially different capability boundaries, such that the same problem may be easy for a stronger reasoner but difficult for a weaker one. Consequently, a difficulty criterion calibrated for one model or benchmark does not necessarily transfer faithfully to another. Applying a shared difficulty standard across reasoners therefore risks conflating task difficulty with model capability, whereas properly calibrating difficulty would require model- and benchmark-specific estimation. Moreover, TH2T requires constructing additional training data and post-training each target reasoner, further increasing the adaptation cost.

MetaCtrl avoids these requirements by estimating how much further computation is useful directly from the evolving reasoning trajectory of the current reasoner. More importantly, the controller is trained once as a smaller meta-level model and can then regulate multiple larger frozen reasoners without retraining them individually. This substantially reduces the cost of deploying efficient reasoning across a family of models: rather than separately collecting data and post-training every target reasoner, a single reusable controller can improve the inference efficiency of multiple larger models. Empirically, on DeepSeek-R1-Distill-Qwen-32B, the same controller achieves 62.1% average accuracy with a 45.6% token reduction, compared with 61.3% and 36.4% for TH2T. This decoupling enables a single controller to generalize across reasoners, eliminating the need for model-specific retraining of computation control.

Output-based methods. Output-based methods regulate computation during inference and are therefore more adaptive than static prompting. However, existing approaches typically rely on predefined stopping criteria, confidence estimates, agreement statistics, or privileged model-internal signals. MetaCtrl instead learns its intervention policy directly from trajectory-level correctness and efficiency outcomes and requires only the observable reasoning trace. On Qwen3-8B, ASAG, the strongest output-based baseline, obtains 69.9% average accuracy with a 31.2% token reduction, whereas MetaCtrl achieves 70.5% accuracy while reducing generation by 50.0%. On Qwen3-14B, MetaCtrl achieves comparable accuracy to ASAG (74.1% vs. 74.0%) but more than doubles the token reduction, from 26.9% to 53.7%. Compared with DEER and Dynasor, MetaCtrl likewise achieves higher average accuracy together with substantially larger reductions in generated tokens. These results indicate that learning trajectory-dependent interventions can provide more aggressive computation savings without relying on manually specified control signals. More broadly, the fixed criteria underlying many output-based methods may not transfer uniformly across reasoners with different capability boundaries or across problems with different computational demands. MetaCtrl instead conditions each control decision on the evolving reasoning trajectory, allowing the regulation strategy to adapt dynamically to both the current reasoner and its problem solving progress.

External reasoning control. ACTS is the most closely related baseline because it also separates the controller from a frozen reasoner and makes online decisions from the evolving trajectory. However, ACTS requires supervised steering data and explicitly conditions the controller on a predefined remaining token budget. MetaCtrl removes both requirements: it is trained directly from outcome-level reinforcement and determines how much additional reasoning is useful without a user-specified budget. The performance gap is consistent across all four evaluated reasoners. On DeepSeek-R1-Distill-Qwen-7B, MetaCtrl improves average accuracy from 52.3% to 59.0% and increases token reduction from 21.8% to 47.3%. On the unseen 32B reasoner, it improves accuracy from 57.8% to 62.1% while increasing token reduction from 11.7% to 45.6%. Similar gains are observed on Qwen3-8B and Qwen3-14B, where MetaCtrl improves average accuracy over ACTS by 7.4 and 7.6 percentage points, respectively, while also achieving larger token reductions (50.0% vs. 44.1% and 53.7% vs. 39.5%). Although increasing the prescribed budget in ACTS may improve accuracy, the minimum budget sufficient for a correct solution is unknown a priori and varies across problems and reasoners, leaving users to choose it heuristically. MetaCtrl instead learns to allocate additional computation online from the evolving reasoning trajectory without requiring such a predefined budget.

Table 6: Performance and reasoning efficiency across unseen reasoners on AMC and LiveCodeBench.

Methods	AMC			LiveCodeBench		
	Acc.↑	Len.↓	Reduc.↑	Acc.↑	Len.↓	Reduc.↑
DeepSeek-R1-Distill-Qwen-7B [Seen Reasoner]						
Vanilla	73.5	7279		38.4	10,516	
MetaCtrl (ours)	74.7	3036	-58.3%	42.3	2,932	-72.1%
Qwen3-8B [Unseen Reasoner]						
Vanilla	78.8	9083		64.7	8,923	
MetaCtrl (ours)	81.9	4623	-49.1%	65.0	5855	-34.4%
Qwen3-14B [Unseen Reasoner]						
Vanilla	81.9	8505		72.6	8,112	
MetaCtrl (ours)	84.3	3938	-53.7%	74.7	5416	-33.2%
DeepSeek-R1-Distill-Llama-8B [Unseen Reasoner]						
Vanilla	67.5	7982		41.5	11029	
MetaCtrl (ours)	68.7	3357	-57.9%	43.1	3861	-65.0%

Table 7: Results of DeepSeek-R1-Distill-Llama-8B across five reasoning benchmarks. **Acc.** denotes pass@1 accuracy, **Len.** denotes the average total generation length, and **Reduc.** denotes the relative reduction in generation length compared with vanilla reasoning.

Methods	MATH-500			AIME 2024			OlympiadBench			GPQA Diamond		
	Acc.↑	Len.↓	Reduc.↑	Acc.↑	Len.↓	Reduc.↑	Acc.↑	Len.↓	Reduc.↑	Acc.↑	Len.↓	Reduc.↑
DeepSeek-R1-Distill-Llama-8B [Unseen Reasoner]												
Vanilla	87.6	3928		40.0	13472		47.3	8416		26.3	9655	
MetaCtrl (ours)	89.0	1942	-50.6%	40.0	7642	-43.3%	50.1	3819	-54.6%	37.4	3192	-66.9%

Across the four paradigms, the results reveal complementary limitations of existing approaches: prompt-based methods decide computation too early, model-based methods require modifying each target reasoner, output-based methods rely on predefined control signals, and existing external controllers require additional supervision or explicit reasoning budgets. MetaCtrl instead learns a separate, budget-free, trajectory-conditioned control policy directly from correctness and efficiency outcomes while keeping the reasoner frozen. Rather than requiring a predefined computation budget, it dynamically determines how much additional reasoning is needed based on the evolving trajectory. This design yields a better accuracy–efficiency trade-off on the training-time reasoner and, critically, transfers to unseen reasoners without additional post-training.

E Additional Experimental Results and Analysis

E.1 Generalization Across Benchmarks, Domains, and Reasoning Model Architectures.

We further evaluate MetaCtrl on additional benchmarks, the code reasoning domain, and an unseen reasoner architecture. As shown in Tables 6 and 7, the same controller transfers to unseen reasoners on AMC and LiveCodeBench without additional training, reducing generation length by 33.2% to 65.0% while maintaining or improving accuracy in all settings. This transfer extends to DeepSeek-R1-Distill-Llama-8B, whose Llama backbone differs from the Qwen-based reasoner used for controller training. On AMC and LiveCodeBench, MetaCtrl improves accuracy from 67.5% to 68.7% and from 41.5% to 43.1%, while reducing generation length by 57.9% and 65.0%, respectively. Across MATH-500, AIME 2024, OlympiadBench, and GPQA Diamond, it further reduces generation length by 43.3% to 66.9% while matching or improving Vanilla accuracy, including an 11.1 percentage point gain on GPQA Diamond. Together, these results show that MetaCtrl is not tied to the training benchmark, reasoning domain, or reasoner architecture, but transfers effectively across unseen tasks and reasoning models without additional training.

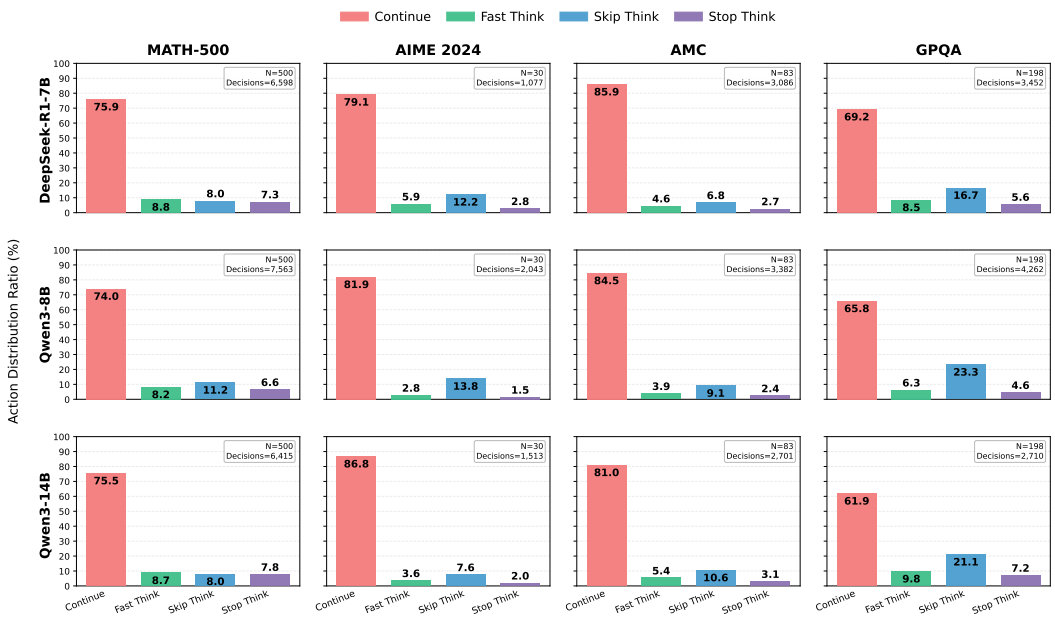


Figure 6: Controller action distributions across different reasoners and benchmarks. Each row corresponds to a reasoner and each column to a benchmark. The learned policy exhibits distinct task–reasoner-specific control patterns rather than a uniform intervention strategy. DeepSeek-R1-7B is used as shorthand for DeepSeek-R1-Distilled-Qwen2.5-7B.

E.2 Adaptive Control Emerges from Task–Reasoner Interactions.

Adaptive control across tasks and reasoners. Figure 6 shows that MetaCtrl maintains a selective control policy across 12 reasoner and benchmark combinations. *Continue* remains the dominant action, accounting for 61.9% to 86.8% of decisions, while *Stop Think* is used only 1.5% to 7.8% of the time. *Skip Think* is more frequent than *Stop Think* in every setting and is the most common intervention action in 10 of the 12 combinations. These statistics indicate that the controller more frequently applies local reasoning compression than explicit termination at individual control points. At the same time, intervention rates vary systematically across tasks. GPQA elicits substantially more intervention than AIME 2024 and AMC across all three reasoners, while MATH 500, despite being relatively easier, consistently shows lower *Continue* rates than AIME 2024 and AMC. This suggests that the controller is not governed by benchmark difficulty alone, but adapts to task specific reasoning patterns, intervening more when shorter or locally compressed reasoning may suffice and preserving longer reasoning when sustained computation remains useful. Overall, these patterns indicate that MetaCtrl largely preserves the reasoner's native computation while adapting both the degree and form of intervention across tasks and reasoners.

Control demand is not explained by task difficulty or trajectory length alone. GPQA consistently induces the highest intervention rates across all three reasoners, reaching 30.8%, 34.2%, and 38.1%, compared with 20.9%, 18.1%, and 13.2% on AIME 2024. Importantly, this difference cannot be attributed simply to longer reasoning trajectories. Although all three reasoners receive substantially more controller decisions per problem on AIME 2024 than on GPQA, the controller intervenes considerably less often on AIME. These results suggest that control demand is not determined by task difficulty or trajectory length alone. Instead, the learned policy appears sensitive to the evolving reasoning trajectory, preserving continued computation in some settings while intervening more frequently in others.

Control demand depends on the model-task pair, not model scale alone. Increasing the reasoner size from Qwen3-8B to Qwen3-14B does not produce a consistent change in intervention rate. It decreases from 18.1% to 13.2% on AIME 2024, but increases from 34.2% to 38.1% on GPQA and from 15.5% to 19.0% on AMC. Meanwhile, MATH-500 remains nearly unchanged

Table 8: Ablation studies on intervention position, action space, and controller initialization. The reasoning model is DeepSeek-R1-Distill-Qwen-7B.

Ablation Setting		MATH-500			AMC			AIME2024			OlympiadBench			GPQA Diamond		
Component	**Setting**	**Acc.**	**Len.**	**Reduc.**	**Acc.**	**Len.**	**Reduc.**	**Acc.**	**Len.**	**Reduc.**	**Acc.**	**Len.**	**Reduc.**	**Acc.**	**Len.**	**Reduc.**
Vanilla (No Controller)		85.2	2857		73.5	7279		50.0	10570		48.6	8347		32.8	5349	
Intervention Position	At Transition Words	71.2	**999**	-65.0%	47.7	**1236**	-83.0%	22.7	**2828**	-73.2%	32.4	**1315**	-84.2%	35.5	**1069**	-80.0%
	Every 300 Tokens	86.0	3638	+27.3%	68.0	9351	+28.5%	41.3	15500	+46.6%	50.7	9587	+14.9%	29.0	15362	+187.2%
Intervention Action	Add Check Action	87.2	2025	-29.1%	70.8	4000	-45.0%	42.7	8793	-16.8%	51.1	3888	-53.4%	38.2	3746	-30.0%
Controller Model	Initialized from Qwen3-1.7B	**88.0**	2295	-19.7%	72.5	4850	-33.4%	41.3	10179	-3.7%	51.6	4946	-40.7%	40.9	4600	-14.0%
MetaCtrl		**88.0**	1644	-42.5%	**74.7**	3036	-58.3%	**56.7**	5899	-44.2%	**52.3**	2965	-64.5%	**42.4**	2234	-58.2%

Table 9: End-to-end inference latency under Vanilla Reasoning and MetaCtrl Controlled Reasoning with different GPU memory allocations. All experiments are conducted on NVIDIA H20 GPUs. **Mem. Frac. (R+C)** denotes the configured static GPU memory fractions for the reasoner (R) and controller (C), respectively. Vanilla Reasoning uses a single GPU for the reasoner. For MetaCtrl with separate GPUs, the reasoner and controller run on two dedicated GPUs. For MetaCtrl with a shared GPU, the reasoner and controller are colocated on the same GPU.

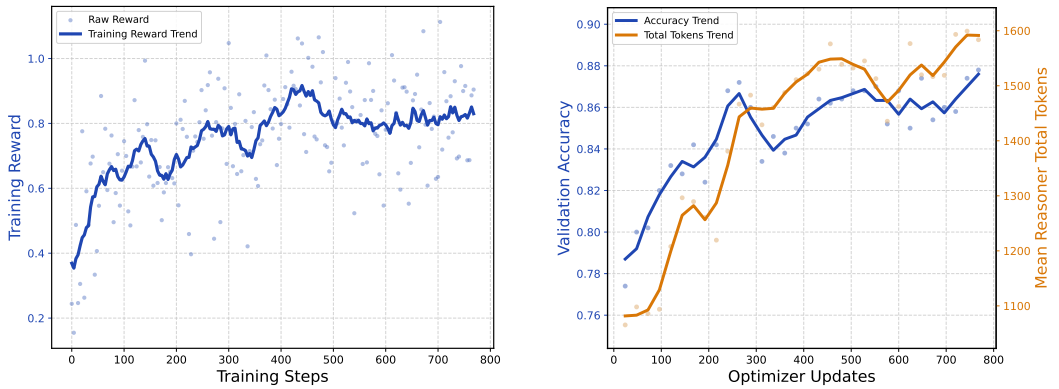
Reasoner	**MATH500**		**AMC**		**GPQA Diamond**	
	Avg. Time (s)	**Mem. Frac. (R+C)**	**Avg. Time (s)**	**Mem. Frac. (R+C)**	**Avg. Time (s)**	**Mem. Frac. (R+C)**
Vanilla Reasoning						
Qwen3-8B	31.62	$0.90 + 0.00$	74.93	$0.90 + 0.00$	81.28	$0.90 + 0.00$
Qwen3-14B	45.18	$0.90 + 0.00$	100.94	$0.90 + 0.00$	95.79	$0.90 + 0.00$
MetaCtrl Controlled Reasoning (Two Separate GPUs)						
Qwen3-8B	15.44	$0.90 + 0.60$	43.02	$0.90 + 0.60$	23.96	$0.90 + 0.60$
Qwen3-14B	20.63	$0.90 + 0.60$	39.57	$0.90 + 0.60$	36.74	$0.90 + 0.60$
MetaCtrl Controlled Reasoning (Two Separate GPUs)						
Qwen3-8B	16.21	$0.90 + 0.40$	38.39	$0.90 + 0.40$	23.77	$0.90 + 0.40$
Qwen3-14B	20.98	$0.90 + 0.40$	38.79	$0.90 + 0.40$	36.95	$0.90 + 0.40$
MetaCtrl Controlled Reasoning (Two Separate GPUs)						
Qwen3-8B	15.71	$0.90 + 0.18$	37.32	$0.90 + 0.18$	24.25	$0.90 + 0.18$
Qwen3-14B	21.21	$0.90 + 0.18$	46.93	$0.90 + 0.18$	39.69	$0.90 + 0.18$
MetaCtrl Controlled Reasoning (One Shared GPU)						
Qwen3-8B	15.95	$0.72 + 0.18$	37.83	$0.72 + 0.18$	21.74	$0.72 + 0.18$
Qwen3-14B	20.79	$0.72 + 0.18$	48.16	$0.72 + 0.18$	33.33	$0.72 + 0.18$

across DeepSeek-R1-7B, Qwen3-8B, and Qwen3-14B, with intervention rates of 24.1%, 26.0%, and 24.5%, respectively. These non-monotonic trends suggest that control demand depends on the interaction between the reasoner and the task rather than on model scale alone. This motivates adaptive external control, since a fixed token budget or uniform stopping rule cannot directly account for such variation across reasoner-task combinations.

Overall, these results provide behavioral evidence for the central motivation of adaptive, budget-free reasoning control: useful computation should be regulated according to the evolving reasoning state, rather than predetermined from model size, task difficulty, or elapsed computation.

E.3 More Ablation Studies

Controller design. Table 8 examines the effects of intervention timing, action space, and controller initialization. Heuristic intervention schedules lead to substantially different tradeoffs. Intervening at transition words produces aggressive compression of 65.0% to 84.2%, but substantially reduces accuracy on most benchmarks, whereas intervening every 300 tokens increases generation length on all five benchmarks and underperforms MetaCtrl in accuracy. These results support conditioning intervention timing on the evolving reasoning state rather than using fixed heuristics. Adding a `Check` action also degrades both accuracy and length reduction relative to the original action space across all benchmarks, suggesting that a larger action space does not necessarily improve control. Similarly, initialization from Qwen3-1.7B results in weaker accuracy and substantially less compression than the default controller on most benchmarks. Overall, the full MetaCtrl design provides the most



(a) Training reward throughout GRPO optimization.

(b) Validation accuracy and average total generation tokens during training.

Figure 7: **Training dynamics of the MetaCtrl.** (a) Training reward during GRPO optimization. (b) Validation accuracy and average total generation tokens throughout training. The reward and accuracy progressively improve and stabilize, while generation length converges to a relatively stable range, indicating that the controller learns to regulate reasoning computation without uniformly compressing reasoning trajectories.

consistent balance between accuracy and generation length, supporting adaptive intervention timing, a compact action space, and the chosen controller initialization.

F The Analysis of Latency and Computation Cost of Methods

End to end latency. Table 9 shows that MetaCtrl consistently reduces end to end inference latency across both reasoners and all three benchmarks. When the reasoner and controller are placed on two separate GPUs, MetaCtrl reduces mean latency by **42.6% to 70.8%** relative to Vanilla Reasoning across the evaluated memory allocations. This indicates that the reduction in reasoner computation is large enough to offset the additional controller inference and communication overhead. Moreover, reducing the controller memory fraction from 0.60 to 0.18 largely preserves the latency improvement, suggesting that MetaCtrl does not require a heavily provisioned controller to achieve substantial acceleration. The shared GPU setting provides further evidence of practical efficiency. When the reasoner and controller share a single H20 with memory fractions of 0.72 and 0.18, respectively, MetaCtrl reduces latency by **49.5% to 73.3%** across the six reasoner and benchmark combinations, with an average reduction of **57.3%**. On GPQA Diamond, for example, latency decreases from 81.28s to 21.74s for Qwen3 8B and from 95.79s to 33.33s for Qwen3 14B. These results show that reductions in reasoning computation translate into substantial wall clock speedups, including when the reasoner and controller are colocated on a single GPU. MetaCtrl does introduce an additional controller model, which increases memory requirements and system complexity compared with Vanilla Reasoning. Nevertheless, the shared GPU results show that this overhead can be reduced substantially while retaining large latency gains.

G Training Dynamics and Efficiency

Stable policy improvement during training. We analyze the optimization dynamics of our MetaCtrl by tracking the training reward throughout GRPO training (Figure 7a). The reward increases rapidly during the early stage of training and continues to improve despite local fluctuations, reaching a substantially higher level after roughly 400 training steps. It subsequently stabilizes within a relatively narrow range, indicating that the controller progressively learns more effective intervention policies rather than relying on transient high-reward behaviors. The sustained improvement and stable late-stage reward further suggest that the proposed optimization objective provides a reliable learning signal for regulating the reasoning process.

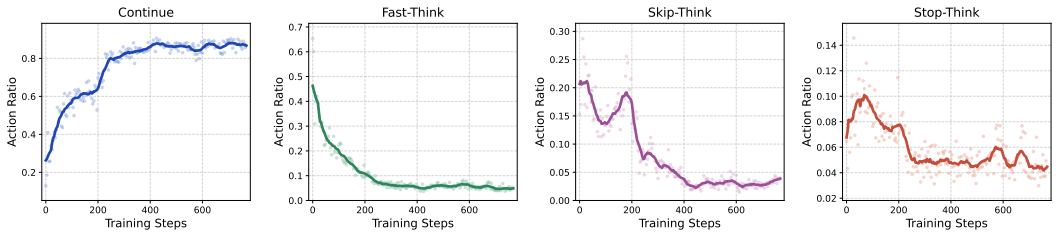


Figure 8: **Evolution of the controller action distribution during training.** The learned policy gradually shifts toward CONTINUE, while the frequencies of FAST-THINK, SKIP-THINK, and STOP-THINK decrease and stabilize at lower levels. This trend suggests that the controller progressively learns to intervene more selectively rather than frequently modifying the reasoner's trajectory.

Balancing accuracy and reasoning length. We track validation accuracy and mean reasoner generation length throughout training in Figure 7b. Validation accuracy increases from approximately 79% at the beginning of training to around 87% in later stages, while mean reasoning length rises from roughly 1.1K to around 1.55K tokens. The two metrics exhibit broadly aligned trends during optimization, indicating that training does not collapse to a trivial policy that simply minimizes reasoning length. Instead, the controller learns to retain additional computation when it is useful for correctness, rather than aggressively compressing reasoning at the expense of accuracy.

Evolution of the controller policy. We further investigate how the learned control strategy evolves during training by tracking the distribution of the four controller actions in Figure 8. At the beginning of training, the controller frequently modifies the reasoner's trajectory through FAST-THINK, SKIP-THINK, and STOP-THINK, while CONTINUE constitutes only a relatively small fraction of the decisions. As optimization proceeds, the action distribution changes markedly: the proportion of CONTINUE steadily increases and eventually dominates the policy, whereas FAST-THINK and SKIP-THINK decrease substantially and STOP-THINK remains comparatively infrequent. Notably, the intervention actions do not vanish completely, but converge to small yet non-zero frequencies. This behavior suggests that the controller does not learn to indiscriminately shorten the frozen reasoner's reasoning process. Instead, it progressively learns to preserve the reasoner's native trajectory in most cases and intervene selectively when modifying the ongoing reasoning is useful. Together with the improvements in reward and accuracy and the stabilization of generation length, these dynamics indicate the emergence of a stable and selective computation policy: the controller learns not only how to intervene, but also when intervention is unnecessary.

H FUTURE WORK

A promising direction is to extend MetaCtrl to a collaborative multi-reasoner system, where multiple reasoning models constitute the object-level and a shared meta-level controller coordinates the overall reasoning process. For each problem, the controller could first perform joint model routing and initial effort allocation, selecting which reasoner should begin solving the problem and under which reasoning effort configuration.

As reasoning unfolds, the controller could continuously monitor the evolving explicit reasoning trace and decide whether to let the current reasoner proceed, intervene through lightweight textual guidance to regulate its subsequent reasoning, or selectively invoke a stronger model for targeted guidance, verification, or correction. Routine reasoning could therefore remain with smaller and less expensive models, while stronger models are reserved for stages where additional capability is most valuable. The resulting system would jointly control computation at two levels: across models, by deciding which reasoner should handle each stage of the solution process, and within a selected reasoner, by regulating how its ongoing reasoning should proceed through trajectory-level interventions.

This setting raises an important resource allocation problem: when a difficult intermediate state should be handled by further regulating the current reasoner, and when it should instead be escalated to a more capable model, selectively invoked at critical stages to provide targeted guidance or

correction. Learning this trade-off online could reduce the need for users to manually choose among model families and reasoning-effort settings, while avoiding the cost of running the strongest model throughout the entire solution process. More broadly, such a system would extend metacognitive control from regulating a single reasoning trajectory to coordinating capability and computation across a collaborative reasoning system, with the goal of achieving a desired level of reliability at lower end-to-end inference cost.

I Case Study

I.1 Preventing Post-Solution Overthinking

As shown in Figures 9–18, across five representative reasoning problems, vanilla reasoning often continues with redundant verification or repeated hypothesis exploration after the key solution has already been established. In contrast, MetaCtrl adaptively regulates the evolving trajectory, preserving necessary derivations while accelerating or terminating further computation when sufficient evidence has accumulated. The consistent preservation of correct answers across these cases illustrates how MetaCtrl reduces diverse forms of overthinking without aggressively truncating reasoning.

I.2 Reducing Errors from Prolonged Reasoning

As shown in Figures 19–24, these cases further illustrate that longer reasoning is not necessarily more reliable. Vanilla reasoning can become trapped in prolonged hypothesis exploration or increasingly complex derivations, where additional deliberation may compound inconsistencies and ultimately lead to incorrect answers. In contrast, MetaCtrl regulates how the frozen reasoner proceeds through lightweight interventions, encouraging more direct reasoning and terminating unnecessary deliberation once a sufficient solution path has emerged. In these examples, such controlled trajectories are not only substantially shorter but also reach the correct answers, suggesting that simpler and more focused reasoning can sometimes be more reliable than unconstrained overthinking.

Case Study 1: MATH-500 ID: 207

Question: Let z be a complex number such that z^5 = 1 and z ≠ 1. Compute z + 1/z + z^2 + 1/z^2.
Answer: -1

Vanilla Reasoning Total Token Number: 7043

Okay, so I need to compute the value of z + 1/z + z^2 + 1/z^2, where z is a complex number satisfying z^5 = 1 and z ≠ 1. Hmm,
[... omitted]
First, since z^5 = 1, that means z is a 5th root of unity. The 5th roots of unity are the solutions to the equation z^5 - 1 = 0. These can be written in the form e^(2πik/5) for k = 0, 1, 2, 3, 4. But since z ≠ 1, we know that z is one of the primitive 5th roots of unity, so k = 1, 2, 3, 4.
[... omitted]
Alternatively, maybe there's a way to express this in terms of z + z^2 + z^3 + z^4 or something like that? Wait, since z^5 = 1, we know that 1 + z + z^2 + z^3 + z^4 = 0. That's a standard identity for roots of unity. So if I can relate the given expression to that sum.\n\nLet me write down the given expression again: z + 1/z + z^2 + 1/z^2. Let me note that 1/z = z^4, because z^5 = 1, so z^5 = 1 implies z^4 = z^(-1). Similarly, 1/z^2 = z^3. Therefore, substituting these into the expression:\n\nz + z^4 + z^2 + z^3.\n\nSo the original expression simplifies to z + z^2 + z^3 + z^4. But from the identity 1 + z + z^2 + z^3 + z^4 = 0, we can subtract 1 from both sides to get z + z^2 + z^3 + z^4 = -1. Therefore, the value of the original expression is -1.\n\nWait, that seems straightforward. Let me check if I did that correctly. Let me verify each step again.\n\nFirst, since z^5 = 1, then z ≠ 0, so 1/z = z^(-1). Also, since z^5 = 1, z^(-1) = z^4, because multiplying both sides by z^(-1), we get z^4 = z^(-1). Similarly, z^(-2) = z^3. Therefore, substituting into the original expression:\n\nz + z^4 + z^2 + z^3. Which is indeed z + z^2 + z^3 + z^4. Then, since 1 + z + z^2 + z^3 + z^4 = 0, subtracting 1 gives z + z^2 + z^3 + z^4 = -1. Therefore, the answer is -1.
[... omitted]
But wait, the problem says "Compute z + 1/z + z^2 + 1/z^2", and according to this, it's equal to -1. But let me check with a specific example. Let me pick a specific 5th root of unity and compute the value.\n\nTake z = e^(2πi/5). Then z^5 = 1. Let me compute z + 1/z + z^2 + 1/z^2.\n\nFirst, 1/z = z^4, and 1/z^2 = z^3. Therefore, the expression is z + z^4 + z^2 + z^3, which is the same as z + z^2 + z^3 + z^4, which we already said is -1. Let me compute this numerically.
[... omitted]
But let me think again if there's another way to approach this problem, just to confirm.\n\nAlternatively, maybe using algebra. Let me denote S = z + 1/z + z^2 + 1/z^2. Let me note that z^5 = 1, so z ≠ 0, so we can write 1/z = z^4, as before. Therefore, S = z + z^4 + z^2 + z^3. Which is the same as (z + z^4) + (z^2 + z^3). Let me compute each pair.
[... omitted]
Alternatively, since z^5 = 1, then z^5 - 1 = 0, which factors as (z - 1)(z^4 + z^3 + z^2 + z + 1) = 0. Since z ≠ 1, we have z^4 + z^3 + z^2 + z + 1 = 0, so z^4 + z^3 + z^2 + z = -1, which is the same as before. Therefore, S = z + z^2 + z^3 + z^4 = -1.
[... omitted]
But just to make sure, let me try another specific example. Let me take z = e^(4πi/5), which is another 5th root of unity. Then, compute z + 1/z + z^2 + 1/z^2. Again, 1/z = z^3, 1/z^2 = z^2. Wait, no: z^5 = 1, so 1/z = z^(5 - 1) = z^4, so 1/z^2 = z^(5 - 2) = z^3. Therefore, same as before, the expression is z + z^4 + z^2 + z^3, which is again the same sum. Therefore, regardless of which root we take (as long as it's not 1), the sum is -1. Therefore, the answer is definitely -1.
[... omitted]
Alternatively, maybe I can compute the expression using trigonometric identities. Let me recall that z = e^(iθ), where θ = 2π/5, so z + 1/z = 2cos(θ), and z^2 + 1/z^2 = 2cos(2θ). Therefore, the expression becomes 2cos(θ) + 2cos(2θ). Let me compute this.\n\nSo 2cos(2π/5) + 2cos(4π/5). Let me compute these values numerically. As before, cos(2π/5) ≈ 0.3090, cos(4π/5) ≈ -0.8090. Therefore, 2*(0.3090) + 2*(-0.8090) = 0.618 - 1.618 = -1. So again, same result. Therefore, this confirms that the value is -1.
[... omitted]
Therefore, through multiple approaches—using properties of roots of unity, trigonometric identities, and algebraic manipulations—we consistently arrive at the conclusion that the value of z + 1/z + z^2 + 1/z^2 is -1.\n\nI think this is thorough enough. I considered different methods, checked with specific examples, and verified through algebraic identities. Therefore, I'm confident that the answer is -1.\n\n**Final Answer**\n\\boxed{-1}\n</think>
To evaluate the expression \n$$\nz + \\frac{1}{z} + z^2 + \\frac{1}{z^2}\n$$ \ngiven that $ z^5 = 1 $ and $ z \\neq 1 $, we begin by recognizing that $ z $ is a **nontrivial 5th root of unity**. The 5th roots of unity are the complex numbers satisfying
[... omitted]
is exactly the same as the sum we found earlier:\n$$\nz + z^2 + z^3 + z^4 = -1.\n$$\n\n---\n\n### Step 3: Final Answer\n\nThus, the value of the original expression is:\n$$\n\\boxed{-1}\n$$

Correct but Inefficient Reasoning

Figure 9: Vanilla reasoning on a MATH-500 example. The model obtains the correct answer but continues with repetitive verification and alternative derivations, leading to substantial overthinking and 7,043 reasoning tokens.

Case Study 1: MATH-500 ID: 207

Intervene Action: Continue → Fast Think → Continue → Continue → Continue → Stop Think

MetaCtrl Reasoning Total Token Number: 957

Okay, so I need to compute the value of z + 1/z + z^2 + 1/z^2 where z is a complex number satisfying z^5 = 1 and z ≠ 1. Hmm, let me think about how to approach this.[Continue]
First, since z^5 = 1, that means z is a 5th root of unity. The 5th roots of unity are the solutions to the equation z^5 - 1 = 0. These can be written in the form e^(2πik/5) for k = 0, 1, 2, 3, 4. But since z ≠ 1, we know that z is one of the primitive 5th roots of unity, so k = 1, 2, 3, 4.
Let's keep only the essential derivation, avoid repetition, and proceed efficiently. Since z^5 = 1, we can use this to simplify expressions involving powers of z.[Continue]
The expression we need to compute is z + 1/z + z^2 + 1/z^2. Let me note that 1/z can be written as z^4 because z^5 = 1 implies z^5 - 1 = 0, so z ≠ 0, and thus 1/z = z^(-1) = z^(5 - 1) = z^4. Similarly, 1/z^2 = z^(-2) = z^(5 - 2) = z^3. Therefore, the expression simplifies to:
z + z^4 + z^2 + z^3.[Continue]
So, the problem reduces to computing z + z^2 + z^3 + z^4.[Continue]
Wait, but I remember that the sum of all 5th roots of unity is 0. That is, 1 + z + z^2 + z^3 + z^4 = 0. Therefore, if we subtract 1 from both sides, we get z + z^2 + z^3 + z^4 = -1. Therefore, the expression we need is equal to -1?
The reasoning is sufficient; state only the final conclusion briefly and close the reasoning now. Therefore, the value of the given expression is \\boxed{-1}.\n\n**Final Answer**\n\ \boxed{-1}\n</think>
We are given that z is a complex number such that z^5 = 1 and z ≠ 1. This tells us that z is a non-trivial 5th root of unity. The 5th roots of unity are the solutions to the equation:\n\nz^5 = 1\n\nThese roots are:\n\nz = e^(2πik/5) for k = 0, 1, 2, 3, 4\n\nSince z ≠ 1, we are considering the roots z = e^(2πik/5) for k = 1, 2, 3, 4.\n\n---\n\nWe are asked to compute the expression:\n\nz + 1/z + z^2 + 1/z^2\n\nLet's simplify this expression using the fact that z^5 = 1. Since z ≠ 0, we can write:\n\n1/z = z^(-1) = z^(5 - 1) = z^4\n1/z^2 = z^(-2) = z^(5 - 2) = z^3\n\nSo the expression becomes:\n\nz + z^4 + z^2 + z^3\n\nNow, recall that the sum of all 5th roots of unity is:\n\n1 + z + z^2 + z^3 + z^4 = 0\n\nSubtracting 1 from both sides gives:\n\nz + z^2 + z^3 + z^4 = -1\n\nThus, the value of the original expression is:\n\n$$\n\\boxed{-1}\n$$

Correct and Efficient Reasoning

Figure 10: MetaCtrl reasoning on the MATH-500 example. Both methods arrive at the correct answer, while MetaCtrl dynamically regulates the reasoning trajectory through lightweight interventions and terminates once the solution is sufficiently established, reducing the reasoning length from 7,043 to 957 tokens.

Case Study 2: MATH-500 ID: 1227

Question: Six witches and ten sorcerers are at an arcane mixer. The witches have decided to shake hands with each of the sorcerers. None of the witches bothered to shake each other's hands, since they are all good friends already, and the sorcerers all sort of hate each other and did not shake hands with other sorcerers. How many handshakes took place at the mixer?
Answer: 60

Vanilla Reasoning Total Token Number: 2479

Okay, so there's this problem about six witches and ten sorcerers at an arcane mixer. The question is asking how many handshakes took place. Let me try to figure this out step by step. \n\nFirst, let me make sure I understand the problem correctly. There are two groups: witches and sorcerers. The witches are shaking hands with the sorcerers, but not with each other. The sorcerers aren't shaking hands with each other either. So, the only handshakes happening are between witches and sorcerers. \n\nLet me break it down. There are 6 witches and 10 sorcerers. Each witch is shaking hands with each sorcerer. So, if I think about it, each of the six witches is going to shake hands with all ten sorcerers. So, for each witch, there are 10 handshakes. Then, since there are six witches, maybe I multiply 6 by 10? That would give me 60 handshakes. \n\nBut wait, let me check if there's another way to think about this. Maybe using combinations? Like, if we consider the total number of
[... omitted]
groups. So, that would be 6 * 10 = 60. So that seems to confirm the first method. \n\nBut let me make sure there isn't a trick here. The problem says \"the witches have decided to shake hands with each of the sorcerers.\" So, does that mean each witch shakes hands with each sorcerer? That seems to be the case. So, if each of the six witches shakes hands with each of the ten sorcerers, then yes, 6*10=60. \n\nBut let me think again. Sometimes, problems can have nuances. For example, if they had said that each witch shakes hands with some number of sorcerers, but here it's \"each of the sorcerers,\" so maybe
[... omitted]
handshake. So, if we do 6*10, that's the total number of unique pairs. \n\nAlternatively, if we thought of it as total possible handshakes minus the ones that didn't happen. Let me try that approach. \n\nTotal number of people is 6 + 10 = 16. If everyone could shake hands with everyone else, the total number of handshakes would be C(16, 2) which is (16*15)/2 = 120.
[... omitted]
total handshakes that did happen would be total possible handshakes (120) minus the ones that didn't happen (60), which is 60. \n\nSo that also gives 60. \n\nTherefore, regardless of the method, it seems like the answer is 60. \n\nBut let me check again. Maybe I made a mistake in the second approach? Let me verify. \n\nTotal possible handshakes: 16 people, so 16*15/2 = 120. \n\nNumber of handshakes not happening: between witches and between sorcerers. \n\nNumber of witch-witch
[... omitted]
\n\nAlternatively, if there was some other restriction, but the problem says that witches don't shake hands with each other, and sorcerers don't shake hands with each other. So, the only handshakes are between witches and sorcerers. \n\nTherefore, the answer is 60. \n\nBut let me check with smaller numbers to see if my reasoning is correct. \n\nSuppose there is 1 witch and 1 sorcerer. Then, the number of handshakes should be 1. According to the formula, 1*1=1. \n\nIf there are 2 witches and
[... omitted]
Therefore, total handshakes: 3-1=2. Which is correct. \n\nSo, that seems to confirm that the method is correct. \n\nTherefore, applying the same logic to the original problem, the answer is 6*10=60. \n\nTherefore, I think the answer is 60. \n\n**Final Answer**\nThe number of handshakes that took place at the mixer is \\boxed{60}.\n</think>
To determine the number of handshakes that took place at the arcane mixer, we begin by analyzing the given conditions:\n\n- There are **6 witches** and **10 sorcerers**.\n- **Witches only shake hands with sorcerers**.\n- **Witches do not shake
[... omitted]
120\n $$\n\n- **Handshakes that did not occur**:\n - **Witch-witch handshakes**: $\\binom{6}{2} = 15$\n - **Sorcerer-sorcerer handshakes**: $\\binom{10}{2} = 45$\n - **Total non-handshakes**: $15 + 45 = 60$\n\n- **Handshakes that did occur**:\n $$\n 120 - 60 = 60\n $$\n\nThis confirms the earlier result.\n\n---\n\n### Final Answer\n\n$$\n\\boxed{60}\n$$

Correct but Inefficient Reasoning

Figure 11: Vanilla reasoning on a MATH-500 example. The model reaches the correct answer early but continues with alternative counting strategies, repeated verification, and additional sanity checks, resulting in 2,479 reasoning tokens.

Case Study 2: MATH-500 ID: 1227

Intervene Action: Continue → Skip Think → Stop Think

MetaCtrl Reasoning Total Token Number: 409

Okay, so there's this problem about six witches and ten sorcerers at an arcane mixer. The question is asking how many handshakes took place. Let me try to figure this out step by step. [Continue]
First, let me make sure I understand the problem correctly. There are two groups: witches and sorcerers. The witches are shaking hands with the sorcerers, but not with each other. The sorcerers aren't shaking hands with each other either. So, the only handshakes happening are between witches and sorcerers.
Let's solve this directly with the minimum necessary reasoning and move to the answer. So, the problem is about counting the number of handshakes between two groups where members of the same group don't shake hands. The reasoning is sufficient; state only the final conclusion briefly and close the reasoning now. The number of handshakes is the product of the number of witches and the number of sorcerers, since each witch shakes hands with each sorcerer. Therefore, the total number of handshakes is 6 * 10 = 60. The answer is 60.\n\n**Final Answer**\nThe number of handshakes that took place at the mixer is \\boxed{60}.\n</think>
To determine the total number of handshakes at the arcane mixer, we analyze the situation based on the given conditions:\n\n- There are **6 witches** and **10 sorcerers**.\n- **Witches only shake hands with sorcerers**, and **not with other witches**.\n- **Sorcerers do not shake hands with other sorcerers**.\n\nThis means that the only handshakes that occur are between **each witch and each sorcerer**. Since every witch shakes hands with every sorcerer, we can calculate the total number of handshakes by multiplying the number of witches by the number of sorcerers.\n\n$$\n\\text{Total handshakes} = 6 \\times 10 = 60\n$$\n\n---\n\n**Final Answer:** \n$$\n\\boxed{60}\n$$



Correct and Efficient Reasoning

Figure 12: MetaCtrl reasoning on the MATH-500 example. After identifying that all valid handshakes occur between the two groups, the controller skips unnecessary intermediate reasoning and terminates once sufficient information for the solution has been established, reducing the reasoning length from 2,479 to 409 tokens while preserving the correct answer.

Case Study 3: GPQA

Question: Your overhear two chemists talking to each other as they leave a synthetic organic chemistry lab. One asks the other "So, how did it go?" The second chemist replies, "Not well - my compounds are on top of each other." What is the second chemist most likely referring to?
(A) The compounds they are working with have similar boiling points.\n(B) The compounds they are working with are bonding to each other through non-covalent/van der Waals interactions.\n(C) The compounds they are working with have similar polarities.\n(D) The compounds they are working with have similar optical rotations.\n\nPlease write your final answer in the form of \\boxed{A}, \\boxed{B}, \\boxed{C}, or \\boxed{D}
Answer: C

Vanilla Reasoning Total Token Number: 2305

Okay, let's try to figure out this chemistry question. So, the scenario is two chemists talking. One asks, \"So, how did it go?\"
[... omitted]
bonding to each other through these interactions, they might aggregate or form complexes. But would that make them \"on top of each other\"? Maybe if they are in a solution and they stick together, but I'm not sure if that's the common phrase. Maybe not the best fit.\n\nOption C: Similar polarities. Similar polarity could mean that they are not separating well in chromatography, for example. If two compounds have similar polarities, they might have similar retention times on a chromatography column, making them hard to separate. But again, how does that relate to being \"on top of each other\"? Maybe if they are in a layer and can't be separated, like in a liquid-liquid extraction? If they are in the same layer, maybe they
[... omitted]
other\"?\n\nAlternatively, in a distillation setup, if two compounds have similar boiling points (option A), they might form an azeotrope, which is a mixture that boils at a constant temperature, making separation difficult. But again, the phrase \"on top of each other\" is a bit vague.\n\nAnother thought: in a TLC (thin-layer chromatography) or column chromatography, if compounds have similar polarities, they might travel up the plate or column together, so they would be \"on top of each other\" in the same spot. That would make option C a possibility. But then, similar polarities would lead to similar Rf values,
[... omitted]
products are not separating because they are in the same phase. For example, if you have a mixture where the compounds are not immiscible, but that's more about layers.\n\nWait, the answer might be option C: similar polarities. Because if two compounds have similar polarities, they might not separate well in a chromatography column, leading them to be \"on top of each other\" in the same spot. But I need to think about the options again.\n\nAlternatively, option A: similar boiling points.
[... omitted]
they are both in the same layer, but that would be if they are miscible. But again, not sure.\n\nWait, the answer is likely C. Because similar polarities would lead to similar Rf values in TLC, making them overlap. So the chemist is saying that during a separation process, the compounds are overlapping, hence \"on top of each other.\" So the answer is C.\n\n
But let me check the options again. Option C is similar polarities. Option A is similar boiling points. Which is more likely?\n\nIn chromatography, polarity is a key factor. If two compounds have similar polarities, they will have similar retention times. So if they are trying to separate them, they might be overlapping. So the chemist is saying that the compounds are not separating, they are on top of each other. So that would be option C.\n\nAlternatively, if they have similar boiling points, that would affect distillation. But the phrase \"on top of each other\" is more about overlapping in a separation technique. So I think C is the answer.\n\nBut wait, another possibility: if the compounds are in the same layer during extraction. For example, if they are both in the organic layer, but that would be if they are both soluble in the organic solvent. But that's not directly related to polarity. If they have similar polarities, they might both be in the same layer. But that's more about solubility. However, the question is about the compounds being \"on top of each other,\" which might refer to being in the same layer. But that's not one of the options. The options are about boiling points, bonding, polarities, or optical rotations.\n\nSo, if the compounds are in the same layer, maybe they are not separating, but that's not directly one of the options. So perhaps the answer is C, similar polarities, leading to overlapping in chromatography.\n\nAlternatively, if they are in a solution and they are not separating, but that's not matching.\n\nAlternatively, if they are in a reaction and the products are not separating because they are similar. But again, not sure.\n\nAnother angle: the phrase \"on top of each other\" could be a play on words. In chemistry, when you have compounds that are not separating, maybe they are \"on top of each other\" in a chromatogram. So, similar polarities (C) would make them have similar Rf values, so they are on top of each other. So I think C is the answer.\n\nBut let me check the other options again. Option B: bonding through non-covalent interactions. If they are bonding to each other, maybe they are forming a complex, but that's not \"on top of each other.\" Option D: similar optical rotations. That would relate to enantiomers or something, but not sure how that would make them on top of each other.\n\nSo, I think the answer is C. So the answer is \\boxed{C}.\n</think>\n\n
When two chemists are discussing the outcome of an experiment, and one says, *\"my compounds are on top of each other,\"* the phrase is likely a metaphor for a common issue in chemical separation techniques. The key to interpreting this is
[... omitted]
similar polarities, which causes them to **overlap** in a chromatographic or analytical technique. This is the best interpretation of the phrase *\"on top of each other.\"*\n\n$$\n\\boxed{C}\n$$

Correct but Inefficient Reasoning

Figure 13: Vanilla reasoning on a GPQA example. Although the model eventually selects the correct answer, it repeatedly revisits competing explanations and re-evaluates previously considered options, resulting in 2,305 reasoning tokens.

Case Study 3: GPQA

Intervene Action: Continue → Fast Think → Skip Think → Stop Think

MetaCtrl Reasoning Total Token Number: 707

Okay, let's try to figure out this chemistry question. So, the scenario is two chemists talking. One asks, \"So, how did it go?\" and the other replies, \"Not well - my compounds are on top of each other.\" The question is asking what the second chemist is referring to, with options A to D. [Continue]
First, I need to think about what \"compounds are on top of each other\" could mean in a lab setting, especially in synthetic organic chemistry. Let me go through each option.
Let's keep only the essential derivation, avoid repetition, and proceed efficiently. Let me think about each option:\n\nOption A: Similar boiling points. If compounds have similar boiling points, they might be difficult to separate by distillation. But how does that relate to being \"on top of each other\"? Maybe if they're in a distillation setup, but I'm not sure if that's the most direct connection.
Let's solve this directly with the minimum necessary reasoning and move to the answer. Option B: Bonding through non-covalent interactions. This could lead to compounds sticking together, but \"on top of each other\" might imply physical layering rather than bonding. Option C: Similar polarities. Compounds with similar polarities might not separate well in chromatography or other separation techniques. If they have similar polarities, they might elute together from a column, making them \"on top of each other\" in a chromatogram. Option D: Similar optical rotations. This relates to optical activity and would affect how they rotate plane-polarized light, but not directly to their physical arrangement. So, the most likely answer is C, as similar polarities can lead to compounds not separating properly, appearing as if they're on top of each other in a chromatogram or during purification. Therefore, the answer is \\boxed{C}.\n\n\n
The reasoning is sufficient; state only the final conclusion briefly and close the reasoning now. The answer is \\boxed{C}.\n</think>
The phrase \"my compounds are on top of each other\" is a metaphor used in a synthetic organic chemistry context. It suggests that the compounds are not separating properly, likely during a purification or analytical process such as chromatography or distillation.\n\n---\n\nLet's analyze the options:\n\n- **(A) Similar boiling points**: While this can make distillation less effective, it doesn't directly imply that the compounds are \"on top of each other\" in a physical sense. This is a possible issue, but not the most direct interpretation.\n\n- **(B) Bonding through non-covalent/van der Waals interactions**: This would imply that the compounds are interacting with each other, but not necessarily that they are layered or overlapping in a physical sense. This is more about molecular behavior than physical arrangement.\n\n- **(C) Similar polarities**: This is the most relevant. Compounds with similar polarities tend to elute together in chromatography or behave similarly in a solvent, making them appear as if they are \"on top of each other\" in a chromatogram or during separation. This is a common issue in purification and analysis, and the phrase fits well with this interpretation.\n\n- **(D) Similar optical rotations**: This relates to optical activity and is not directly tied to physical layering or separation issues.\n\n---\n\nGiven the context and the phrasing, the best interpretation is that the compounds are not separating properly due to similar polarities, which leads to them appearing as if they are \"on top of each other\" in a chromatographic or purification process.\n\n$$\n\\boxed{C}\n$$

Correct and Efficient Reasoning

Figure 14: MetaCtrl reasoning on the same GPQA example. The controller progressively regulates the reasoning trajectory through Fast Think, Skip Think, and Stop Think interventions, reducing repeated hypothesis exploration and reaching the same correct answer with 707 reasoning tokens.

Case Study 4: AIME2024 ID: 71

Question: Consider the paths of length 16 that follow the lines from the lower left corner to the upper right corner on an 8×8 grid. Find the number of such paths that change direction exactly four times, as in the examples shown below. [... omitted]
Answer: 294

Vanilla Reasoning Total Token Number: 6637

Okay, so I need to find the number of paths of length 16 on an 8x8 grid that start at the lower left corner and end at the upper right corner, changing direction exactly four times. The paths can only move along the grid lines, which I assume means moving either right or up. Let me think about how to approach this.
First, let me recall that in an 8x8 grid, moving from the lower left to the upper right corner requires moving 8 steps to the right and 8 steps up, totaling 16 steps. So any path is a sequence of 8 rights (R) and 8 ups (U).
[... omitted]
So, if a path changes direction exactly four times, that means it has five segments of consecutive moves in the same direction.
[... omitted]
Case 1: Starting with R. Then the direction sequence is R, U, R, U, R. So there are three R segments and two U segments. Similarly, if starting with U, the direction sequence is U, R, U, R, U, so three U segments and two R segments.
[... omitted]
But wait, let me check. For example, if starting with R, the number of compositions of 8 into three positive integers is C(8-1,3-1) = C(7,2) = 21. Similarly, the number of compositions of 8 into two positive integers is C(8-1,2-1) = C(7,1) = 7. Therefore, for starting with R, the number of paths would be 21 * 7. Similarly, starting with U, it would be the same: compositions of 8 into three parts (for U) and two parts (for R). So again 21 * 7. Therefore, total number of paths would be 2 * 21 * 7 = 294.
Wait, but hold on. Is that correct? Let me think again.
Wait, compositions of n into k parts is C(n-1, k-1). So for compositions of 8 into three parts, it's C(7,2) = 21. For compositions of 8 into two parts, it's C(7,1) = 7. So yes, 21*7 *for each case. Then total is* 21*7 = 294.
But wait, the answer seems low. Let me check with an example.
[... omitted]
So according to my calculation, the total number of such paths is 294. But let me check if that's correct.
Alternatively, maybe I need to think about the number of direction changes.
[... omitted]
Therefore, the number of ways to split the R moves into three segments is C(8-1, 3-1) = C(7,2) = 21. Similarly, splitting the U moves into two segments is C(8-1, 2-1) = 7. So 21*7 *for starting with R. Similarly for starting with U,* 21*7. So total 2*21*7 = 294.
But let me check with smaller grids.
[... omitted]
Therefore, total number of paths: 147 + 147 = 294.
Therefore, the answer is 294.
But wait, let me check if there's a possibility that some paths might be counted twice or something. But since starting with R and starting with U are distinct cases, and there's no overlap between them, so adding them is correct.
Alternatively, maybe there's a different approach.
[... omitted]
So total 294.
Therefore, I think this is correct.
But let me check with another approach.
[... omitted]
Therefore, I think the answer is 294.
But let me check if there's a possible mistake in the composition counts.
[... omitted]
So 2*147=294.
Therefore, I think the answer is 294.
But let me check with another example.
[... omitted]
Therefore, applying the same logic to the original problem, the answer is 294.
[... omitted]
Therefore, I think my answer is correct.
But just to be thorough, let me think if there's another way someone might approach this problem and get a different answer.
[... omitted]
Total 294.
Therefore, I think this is correct.
[... omitted]
Therefore, I think the answer is \\boxed{294}.\n\n**Final Answer**\n\\boxed{294}\n</think>
To solve the problem, we consider paths from the lower-left corner $(0,0)$ to the upper-right corner $(8,8)$ on an $8 \\times
[... omitted]
paths that change direction exactly four times:\n\n$$\n147 + 147 = \\boxed{294}\n$$\n\n---\n\n### Final Answer\n\n$$\n\\boxed{294}\n$$

Correct but Inefficient Reasoning

Figure 15: Vanilla reasoning on an AIME 2024 example. The model derives the correct composition-based counting argument but continues to revisit the same calculation through repeated verification and alternative checks, resulting in 6,637 reasoning tokens.

Case Study 4: AIME2024 ID: 71

Intervene Action: Continue → Fast Think → Skip Think → Continue → Continue → Continue → Continue → Stop Think

MetaCtrl Reasoning Total Token Number: 1789

Okay, so I need to find the number of paths of length 16 on an 8x8 grid that start at the lower left corner and end at the upper right corner, changing direction exactly four times. The paths can only move along the grid lines, which I assume means moving either right or up. Let me think about how to approach this. [Continue]
First, let me recall that in an 8x8 grid, moving from the lower left to the upper right corner requires moving 8 steps to the right and 8 steps up, totaling 16 steps. So any path is a sequence of 8 rights (R) and 8 ups (U). The total number of such paths without any restrictions is C(16,8), which is 12870. But here, we have a restriction: the path must change direction exactly four times.
Let's keep only the essential derivation, avoid repetition, and proceed efficiently. So, changing direction exactly four times. Let me think about what that means. Each time the path changes direction, it switches from moving right to up or vice versa. So, for example, a path that goes all right then all up would have only one change of direction. Similarly, a path that alternates directions would have more changes.
Let's solve this directly with the minimum necessary reasoning and move to the answer. So, if a path changes direction exactly four times, then it must consist of five segments. Each segment is either all right or all up. For example, if the path starts with right, then changes to up, then changes to right, etc., alternating directions. Since there are four changes, there are five segments. [Continue]
But the direction of the segments alternates. So, the first segment is either right or up, then the second is the opposite, third is the first again, etc. So, depending on whether the first segment is right or up, the number of right and up segments will differ. However, since the total number of right and up steps is fixed (8 each), we need to ensure that the number of right segments and up segments are such that their total steps add up to 8 each. [Continue]
Let me formalize this. Let's suppose that the path starts with a right move. Then the segments alternate between right and up. Since there are five segments, the number of right segments would be 3 (positions 1, 3, 5) and the number of up segments would be 2 (positions 2, 4). Similarly, if the path starts with an up move, then the number of up segments would be 3 and right segments would be 2.[Continue]
But since the total number of right steps is 8 and up steps is 8, we need to distribute these steps among the segments. Let me consider both cases: starting with right or starting with up. [Continue]
Case 1: Starting with right. Then there are 3 right segments and 2 up segments. Let me denote the lengths of the right segments as r1, r2, r3 and the lengths of the up segments as u1, u2. Then we have:\n\nr1 + r2 + r3 = 8 (total right steps)\nu1 + u2 = 8 (total up steps)\n\nSimilarly, for Case 2: Starting with up. Then there are 3 up segments and 2 right segments:\n\nu1 + u2 + u3 = 8\nr1 + r2 = 8\n\nNow, for each case, the number of ways to distribute the steps into the segments. For example, in Case 1, the number of ways to split 8 right steps into 3 segments is C(8-1, 3-1) = C(7,2) = 21. Similarly, the number of ways to split 8 up steps into 2 segments is C(8-1, 2-1) = C(7,1) = 7. Therefore, the total number of paths starting with right is 21 * 7 = 147. Similarly, for Case 2, starting with up, the number of ways to split 8 up steps into 3 segments is C(7,2) = 21, and splitting 8 right steps into 2 segments is C(7,1) = 7. So total for Case 2 is also 21 * 7 = 147. Therefore, total number of paths is 147 + 147 = 294.
The reasoning is sufficient; state only the final conclusion briefly and close the reasoning now. So, the answer is 294. Therefore, the number of such paths is \\boxed{294}.\n\n**Final Answer**\n\\boxed{294}\n</think>
To determine the number of paths of length 16 on an $8 \\times 8$ grid that start at the lower-left corner and end at the upper-right corner, and change direction **exactly four times**, we begin by analyzing the structure of such paths
[... omitted]
1} = \\binom{7}{1} = 7\n $$\n- Total number of such paths starting with U:\n $$\n 21 \\times 7 = 147\n $$\n\n---\n\n### **Final Calculation**\n\nAdding both cases together:\n\n$$\n147 + 147 = \\boxed{294}\n$$\n\n---\n\n### **Final Answer**\n\n$$\n\\boxed{294}\n$$

Correct and Efficient Reasoning

Figure 16: MetaCtrl reasoning on the same AIME 2024 example. The controller dynamically accelerates the reasoning trajectory and terminates once the composition-based counting argument is complete, reaching the same correct answer with 1,789 reasoning tokens.

Case Study 5: OlympiadBench ID: 2482

Question: Determine all real values of x for which 3^{x+2}+2^{x+2}+2^{x}=2^{x+5}+3^{x}
Answer: 3

Vanilla Reasoning **Total Token Number: 6064**

Okay, so I need to solve this equation: 3^{x+2} + 2^{x+2} + 2^x = 2^{x+5} + 3^x. Hmm, let me
[... omitted]
So the right side is 32*2^x + 3^x.\n\nSo now the equation is:\n\n9*3^x + 5*2^x = 32*2^x + 3^x.\n\nHmm, let me bring all terms to one side to see if I can combine like terms. Subtract 32*2^x and 3^x from both sides:\n\n9*3^x - 3^x + 5*2^x - 32*2^x = 0.\n\nSimplifying each term:\n\nFor the 3^x terms: 9*3^x - 3^x = (9 - 1)*3^x = 8*3^x.\n\nFor the 2^x terms: 5*2^x - 32*2^x = (5 - 32)*2^x = -27*2^x.\n\nSo the equation becomes:\n\n8*3^x - 27*2^x = 0.\n\nHmm, so 8*3^x = 27*2^x. Let me write that as:\n\n8*3^x = 27*2^x.\n\nMaybe I can divide both sides by 2^x to get terms with 3^x and 2^x. Let me try that:\n\n8*(3/2)^x = 27.\n\nWait, because dividing 3^x by 2^x is (3/2)^x. So:\n\n8*(3/2)^x = 27.\n\nThen, divide both sides by 8:\n\n(3/2)^x = 27/8.\n\nNow, 27 is 3^3 and 8 is 2^3, so 27/8 is (3/2)^3. Therefore:\n\n(3/2)^x = (3/2)^3.\n\nSince the bases are the same and positive (and not equal to 1), the exponents must be equal. Therefore, x = 3.\n\nWait, so is that the only solution?Let me check if there are other possibilities. For example, if the base was between 0 and 1, the function would be decreasing, but since 3/2 is greater than 1, the function (3/2)^x is strictly increasing. Therefore, there's only one solution where the exponents are equal. So x = 3.\n\nBut let me verify this solution by plugging it back into the original equation to make sure I didn't make any mistakes during simplification.
[... omitted]
But wait, could there be other solutions? Let me think.
[... omitted]
Wait, let me check with specific numbers.
[... omitted]
Therefore, only crosses zero once at x=3. Therefore, x=3 is the only solution.\n\nTherefore, the answer is x=3.\n\nBut let me check if there's another way to approach the original equation, maybe by substitution.
[... omitted]
Same result.\n\nAlternatively, divide both sides by 2^x:\n\nOriginal equation: 8*3^x - 27*2^x = 0.\n\nDivide by 2^x:\n\n8*(3/2)^x - 27 = 0 => 8*(3/2)^x = 27 => same as before.\n\nSo regardless of the method, x=3 is the only solution.\n\nTherefore, I think that's the only real solution. Let me check once more with another value, say x=4, which we saw gives 216>0, and x=3 gives 0, so it's increasing after x≈1.86. Therefore, only one solution.\n\nTherefore, the answer is x=3.\n\n**Final Answer**\nThe real value of $ x $ is \\boxed{3}.\n</think>
To solve the equation:\n\n$$\n3^{x+2} + 2^{x+2} + 2^x = 2^{x+5} + 3^x\n$$\n\nwe begin by simplifying both sides using the properties of exponents.\n\n---\n\n### Step 1: Simplify both sides\n\n**Left-hand side:**\n\n$$\n3^{x+2} + 2^{x+2} + 2^x = 9 \\cdot 3^x + 4 \\cdot 2^x + 2^x = 9 \\cdot 3^x + 5 \\cdot 2^x\n$$\n\n**Right-hand side:**\n\n$$\n2^{x+5} + 3^x = 32 \\cdot 2^x + 3^x\n$$\n\nSo the equation becomes:\n\n$$\n9 \\cdot 3^x + 5 \\cdot 2^x = 32 \\cdot 2^x + 3^x\n$$\n\n---\n\n### Step 2: Bring all terms to one side\n\n$$\n9 \\cdot 3^x - 3^x + 5 \\cdot 2^x - 32 \\cdot 2^x = 0\n$$\n\n$$\n8 \\cdot 3^x - 27 \\cdot 2^x = 0\n$$\n\n---\n\n### Step 3: Solve the simplified equation\n\n$$\n8 \\cdot 3^x = 27 \\cdot 2^x\n$$\n\nDivide both sides by $ 2^x $:\n\n$$\n8 \\cdot \\left(\\frac{3}{2} \\right)^x = 27\n$$\n\n$$\n\\left(\\frac{3}{2} \\right)^x = \\frac{27}{8}\n$$\n\nNote that $ \\frac{27}{8} = \\left(\\frac{3}{2} \\right)^3 $, so:\n\n$$\n\\left(\\frac{3}{2} \\right)^x = \\left(\\frac{3}{2} \\right)^3\n$$\n\nSince the base $ \\frac{3}{2} > 1 $, the function $ \\left(\\frac{3}{2} \\right)^x $ is strictly increasing. Therefore, the exponents must be equal:\n\n$$\nx = 3\n$$\n\n---\n\n### Step 4: Verify the solution\n\nPlug $ x = 3 $ into the original equation:\n\n**Left-hand side:**\n\n$$\n3^{5} + 2^{5} + 2^3 = 243 + 32 + 8 = 283\n$$\n\n**Right-hand side:**\n\n$$\n2^{8} + 3^3 = 256 + 27 = 283\n$$\n\nBoth sides are equal, confirming that $ x = 3 $ is a solution.\n\n---\n\n### Step 5: Check for other possible solutions\n\nLet's define the function:\n\n$$\nf(x) = 8 \\cdot 3^x - 27 \\cdot 2^x\n$$\n\nWe analyze its behavior:\n\n- As $ x \\to \\infty $, $ 3^x $ dominates, so $ f(x) \\to \\infty $\n- As $ x \\to -\\infty $, both $ 3^x $ and $ 2^x $ approach 0, but $ 2^x $ dominates, so $ f(x) \\to 0^- $\n- The function is continuous and strictly increasing for $ x > \\log_{3/2}(27/8) \\approx 1.86 $, and strictly decreasing before that.\n\nThus, the function crosses zero exactly once at $ x = 3 $, and no other real solutions exist.\n\n---\n\n### Final Answer\n\n$$\n\\boxed{3}\n$$

Correct but Inefficient Reasoning

Figure 17: Vanilla reasoning on an OlympiadBench example. The model correctly derives the unique solution but continues with repeated substitution checks, alternative derivations, and unnecessary uniqueness verification, resulting in 6,064 reasoning tokens.

Case Study 5: OlympiadBench ID: 2482

Intervene Action: Continue → Continue → Fast Think → Stop Think

MetaCtrl Reasoning Total Token Number: 1463

Okay, so I need to solve this equation: 3^{x+2} + 2^{x+2} + 2^x = 2^{x+5} + 3^x. Hmm, let me try to figure this out step by step. First, maybe I can simplify both sides by using exponent rules. Let me recall that a^{b+c} = a^b * a^c. So, let me rewrite each term. [Continue]
Starting with the left side: 3^{x+2} is the same as 3^x * 3^2, which is 9*3^x. Similarly, 2^{x+2} is 2^x * 2^2 = 4*2^x. Then there's another 2^x term. So altogether, the left side becomes 9*3^x + 4*2^x + 2^x. Wait, combining the 4*2^x and 2^x terms, that would be (4 + 1)*2^x = 5*2^x. So left side simplifies to 9*3^x + 5*2^x. [Continue]
Now the right side: 2^{x+5} is 2^x * 2^5 = 32*2^x, and then there's 3^x. So the right side is 32*2^x + 3^x.
Let's keep only the essential derivation, avoid repetition, and proceed efficiently. So, the equation becomes:\n\nLeft side: 9*3^x + 5*2^x\n\nRight side: 32*2^x + 3^x\n\nSo, bringing all terms to one side:\n\n9*3^x + 5*2^x - 32*2^x - 3^x = 0\n\nSimplify terms:\n\nFor 3^x terms: 9*3^x - 3^x = (9 - 1)*3^x = 8*3^x\n\nFor 2^x terms: 5*2^x - 32*2^x = (5 - 32)*2^x = -27*2^x\n\nSo the equation simplifies to:\n\n8*3^x - 27*2^x = 0\n\nWhich can be written as:\n\n8*3^x = 27*2^x\n\nHmm, okay. Let me divide both sides by 2^x to get:\n\n8*(3/2)^x = 27\n\nWait, because 3^x / 2^x is (3/2)^x. So:\n\n8*(3/2)^x = 27\n\nThen, divide both sides by 8:\n\n(3/2)^x = 27/8\n\nNow, 27 is 3^3 and 8 is 2^3, so 27/8 is (3/2)^3. Therefore:\n\n(3/2)^x = (3/2)^3\n\nSince the bases are the same and positive (and not equal to 1), the exponents must be equal. Therefore:\n\nx = 3\n\nWait, so is that the only solution? Let me check if there are other possibilities. For example, if the base was between 0 and 1, the function would be decreasing, but since 3/2 is greater than 1, the function (3/2)^x is strictly increasing. Therefore, there's only one solution where x=3.
The reasoning is sufficient; state only the final conclusion briefly and close the reasoning now. So, the real value of x is 3.\n\n**Final Answer**\nThe real value of $ x $ is \\boxed{3}.\n</think>
To solve the equation:\n\n$$\n3^{x+2} + 2^{x+2} + 2^x = 2^{x+5} + 3^x\n$$\n\nwe begin by simplifying both sides using the exponent rule $ a^{b+c} = a^b \\cdot a^c $.\n\n---\n\n### Step 1: Simplify both sides\n\n**Left-hand side:**\n\n$$\n3^{x+2} = 3^x \\cdot 3^2 = 9 \\cdot 3^x \\\\\n2^{x+2} = 2^x \\cdot 2^2 = 4 \\cdot 2^x \\\\\n\\text{So, left side becomes: } 9 \\cdot 3^x + 4 \\cdot 2^x + 2^x\n$$\n\nCombine like terms:\n\n$$\n9 \\cdot 3^x + (4 + 1) \\cdot 2^x = 9 \\cdot 3^x + 5 \\cdot 2^x\n$$\n\n**Right-hand side:**\n\n$$\n2^{x+5} = 2^x \\cdot 2^5 = 32 \\cdot 2^x \\\\\n\\text{So, right side becomes: } 32 \\cdot 2^x + 3^x\n$$\n\n---\n\n### Step 2: Bring all terms to one side\n\n$$\n9 \\cdot 3^x + 5 \\cdot 2^x - 32 \\cdot 2^x - 3^x = 0\n$$\n\nGroup like terms:\n\n$$\n- $ 9 \\cdot 3^x - 3^x = 8 \\cdot 3^x $\n- $ 5 \\cdot 2^x - 32 \\cdot 2^x = -27 \\cdot 2^x $\n$$\n\nSo the equation becomes:\n\n$$\n8 \\cdot 3^x - 27 \\cdot 2^x = 0\n$$\n\n---\n\n### Step 3: Solve the simplified equation\n\n$$\n8 \\cdot 3^x = 27 \\cdot 2^x\n$$\n\nDivide both sides by $ 2^x $:\n\n$$\n8 \\cdot \\left(\\frac{3}{2} \\right)^x = 27\n$$\n\nDivide both sides by 8:\n\n$$\n\\left(\\frac{3}{2} \\right)^x = \\frac{27}{8}\n$$\n\nNote that $ \\frac{27}{8} = \\left(\\frac{3}{2} \\right)^3 $, so:\n\n$$\n\\left(\\frac{3}{2} \\right)^x = \\left(\\frac{3}{2} \\right)^3\n$$\n\nSince the base $ \\frac{3}{2} > 1 $, the function $ \\left(\\frac{3}{2} \\right)^x $ is strictly increasing, and thus the exponents must be equal:\n\n$$\nx = 3\n$$\n\n---\n\n### Final Answer\n\n$$\n\\boxed{3}\n$$

Correct and Efficient Reasoning

Figure 18: MetaCtrl reasoning on the same OlympiadBench example. After the key algebraic reduction establishes the unique solution, the controller terminates further deliberation, reaching the same correct answer with 1,463 reasoning tokens.

Case Study 6: AMC ID: 41

Question: The figure below depicts a regular 7-gon inscribed in a unit circle. [asy] import geometry; unitsize(3cm); draw(circle((0,0),1),linewidth(1.5)); for (int i = 0; i < 7; ++i) { for (int j = 0; j < i; ++j) { draw(dir(i * 360/7) -- dir(j * 360/7),linewidth(1.5)); } } for(int i = 0; i < 7; ++i) { dot(dir(i * 360/7),5+black); } [/asy] What is the sum of the 4th powers of the lengths of all 21of its edges and diagonals? Output the remainder when the answer is divided by 100.
Answer: 47

Vanilla Reasoning Total Token Number: 8535

Okay, so I have this problem where I need to find the sum of the 4th powers of the lengths of all 21 edges and diagonals of a
[... omitted]
Therefore, the lengths of the edges and diagonals are $2\sin\left(\frac{\pi}{7}\right)$, $2\sin\left(\frac{2\pi}{7}\right)$, and $2\sin\left(\frac{3\pi}{7}\right)$. Each of these lengths occurs multiple times in the 7-gon.
[... omitted]
Since the polygon is regular, each chord length corresponds to a specific step between vertices. For $k = 1$, each vertex connects to the next one, so there are 7 edges of this length. Similarly, for $k = 2$, each vertex connects to the vertex two steps away, which also gives 7 diagonals. The same goes for $k = 3$, giving another 7 diagonals. So, each chord length occurs exactly 7 times.\n\nTherefore, the sum we're looking for is .
[... omitted]
So, Sum = $112\times[sin^4(\pi/7) + sin^4(2\pi/7) + sin^4(3\pi/7)]$
[... omitted]
Let me denote:\n\nA = cos(π/7) + cos(3π/7) + cos(5π/7) \n\nB = cos(2π/7) + cos(4π/7) + cos(6π/7) \n\nI know that in a regular 7-gon, the sum of cos(2πk/7) for k=1 to 6 is -1, so A + B = -1. \n\nAlso, I remember that in some problems, the sum A is known. Let me recall that cos(π/7) + cos(3π/7) + cos(5π/7) = 1/2. Is that correct? Wait, I think that's the case for a regular 7-gon. Let me verify. \n\nYes, actually, I recall that in a regular 7-gon, the sum of cos(π/7) + cos(3π/7) + cos(5π/7) equals 1/2. So, A = 1/2. \n\nTherefore, since A + B = -1, and A = 1/2, then B = -1 - 1/2 = -3/2. \n\nSo, B = cos(2π/7) + cos(4π/7) + cos(6π/7) = -3/2. \n\nWait, that's a useful result.
[... omitted]
So, [sin(3π/7) / sin(π/7)] * cos(4π/7) = -3/2\n\nTherefore, sin(3π/7) * cos(4π/7) = (-3/2) * sin(π/7)
[... omitted]
So, sin(3π/7) cos(4π/7) = [sin(7π/7) + sin(-π/7)] / 2 = [sin(π) + sin(-π/7)] / 2 = [0 - sin(π/7)] / 2 = -sin(π/7)/2\n\nBut from earlier, we have sin(3π/7) * cos(4π/7) = (-3/2) * sin(π/7) \n\nSo, equating the two expressions:\n\n-sin(π/7)/2 = (-3/2) sin(π/7)
[... omitted]
2 sin(π/7) = 0\n\nWait, that can't be right because sin(π/7) is not zero. So, I must have made a mistake somewhere. \n\nWait, let's go back. The sum of cos(2π/7) + cos(4π/7) + cos(6π/7) = -3/2, which we found earlier. \n\nBut when I tried to compute the same sum using the arithmetic progression formula, I ended up with an inconsistency. That suggests that perhaps I made a mistake in applying the formula.
[... omitted]
Which is impossible. So, this suggests that there's a mistake in the initial assumption or in the approach. \n\nWait, perhaps I made a mistake in the initial step when I expressed the sum of sin^4 terms. Let me go back.
[... omitted]
Therefore, C = -3/2\n\nSo, going back to the sum inside the brackets:\n\nSum inside the brackets = 15 + C = 15 + (-3/2) = 15 - 1.5 = 13.5 = 27/2\n\nTherefore, the sum inside the brackets is 27/2. \n\nSo, going back to the total sum:\n\nSum = 112 * (27/2) = 112 * 13.5
[... omitted]
Therefore, the sum of the 21 edge and diagonal lengths raised to the 4th power is 1512. \n\nNow, the problem asks for the remainder when this sum is divided by 100. So, 1512 divided by 100 is 15 with a remainder of 12. .\n\n**Final Answer**\n\\boxed{12}\n</think>
To find the sum of the 4th powers of the lengths of all 21 edges and diagonals of a regular 7-gon inscribed in a unit circle, we start by noting that the lengths of the edges and diagonals are given by \\(2 \\sin\\left(\\frac{k\\pi}{7}\\right)\\) for \\(k = 1, 2,
[... omitted]
Finally, the remainder when 1512 is divided by 100 is:\n\n\\[\n1512 \\mod 100 = 12\n\\]\n\n\\[\n\\boxed{12}\n\\]

Wrong Answer

Figure 19: Vanilla reasoning on an AMC example. The model enters an inconsistent trigonometric detour and continues reasoning despite recognizing contradictions in its intermediate derivation, ultimately producing an incorrect answer after 8,535 reasoning tokens.

Case Study 6: AMC ID: 41

Intervene Action: Continue → Fast Think → Continue → Continue → Continue → Skip Think → Continue → Continue → Continue → Continue → Continue → Continue → Continue → Continue → Continue → Fast Think → Continue → Stop Think

MetaCtrl Reasoning Total Token Number: 2479

Okay, so I have this problem where I need to find the sum of the 4th powers of the lengths of all 21 edges and diagonals of a regular 7-gon inscribed in a unit circle. Then, I have to output the remainder when that sum is divided by 100. Hmm, that sounds a bit complicated, but let me try to break it down.[Continue]
First, I know that in a regular polygon inscribed in a unit circle, the length of each side or diagonal can be determined using the central angle between the two vertices. For a regular 7-gon, the central angles between consecutive vertices are multiples of 360/7 degrees, right? So, the lengths of the edges and diagonals correspond to chords of the circle subtended by angles of k*(360/7) degrees, where k is 1, 2, 3, etc., up to 3 because beyond that, the chords start repeating in the other direction.
Let's keep only the essential derivation, avoid repetition, and proceed efficiently. So, for a regular n-gon inscribed in a unit circle, the length of a chord subtended by an angle θ is given by 2*sin(θ/2). Since we're dealing with a 7-gon, the central angles for the edges and diagonals will be multiples of 2π/7 radians. So, the lengths will be 2*sin(kπ/7) for k = 1, 2, 3. [Continue]
But wait, in a 7-gon, the distinct chord lengths correspond to k = 1, 2, 3 because beyond k=3, the chords start repeating as they become the same as the shorter chords in the opposite direction. So, for each k from 1 to 3, there are 7 chords of each length. Therefore, in total, we have 7 chords for each k=1,2,3, making 21 chords in total, which matches the problem statement. [Continue]
So, the lengths are 2*sin(π/7), 2*sin(2π/7), and 2*sin(3π/7). Each of these lengths occurs 7 times. Therefore, the sum we need is 7*((2*sin(π/7))^4 + (2*sin(2π/7))^4 + (2*sin(3π/7))^4). So, I need to compute this sum and then find its remainder modulo 100. [Continue]
Alright, so let's denote S = 7*((2*sin(π/7))^4 + (2*sin(2π/7))^4 + (2*sin(3π/7))^4). I need to compute S and then find S mod 100.
Let's solve this directly with the minimum necessary reasoning and move to the answer. So, first, let's compute each term (2*sin(kπ/7))^4 for k=1,2,3. [Continue]
We can use the identity for sin^4(x). Remember that sin^4(x) can be expressed in terms of multiple angles. The formula is:\n\nsin^4(x) = (3 - 4*cos(2x) + cos(4x))/8\n\nSo, (2*sin(x))^4 = 16*sin^4(x) = 16*(3 - 4*cos(2x) + cos(4x))/8 = 2*(3 - 4*cos(2x) + cos(4x)) = 6 - 8*cos(2x) + 2*cos(4x)\n\nTherefore, (2*sin(kπ/7))^4 = 6 - 8*cos(2kπ/7) + 2*cos(4kπ/7)\n\nSo, for each k=1,2,3, we can write:\n\n(2*sin(kπ/7))^4 = 6 - 8*cos(2kπ/7) + 2*cos(4kπ/7)\n\nTherefore, the sum over k=1,2,3 of (2*sin(kπ/7))^4 is:\n\nSum = [6 - 8*cos(2π/7) + 2*cos(4π/7)] + [6 - 8*cos(4π/7) + 2*cos(8π/7)] + [6 - 8*cos(6π/7) + 2*cos(12π/7)]\n\nSimplify this:\n\nSum = 6*3 - 8*(cos(2π/7) + cos(4π/7) + cos(6π/7)) + 2*(cos(4π/7) + cos(8π/7) + cos(12π/7))\n\nSo, Sum = 18 - 8*(cos(2π/7) + cos(4π/7) + cos(6π/7)) + 2*(cos(4π/7) + cos(8π/7) + cos(12π/7))\n\nNow, let's compute the sums of cosines. [Continue]
First, let's compute cos(2π/7) + cos(4π/7) + cos(6π/7). I remember that in a regular 7-gon, the sum of these cosines is equal to -1/2. Let me verify that. [Continue]
Yes, in a regular heptagon, the sum of cos(2πk/7) for k=1 to 3 is equal to -1/2. So, cos(2π/7) + cos(4π/7) + cos(6π/7) = -1/2. [Continue]
Similarly, let's compute cos(4π/7) + cos(8π/7) + cos(12π/7). Let's note that 8π/7 is equivalent to 8π/7 - 2π = -6π/7, and 12π/7 is equivalent to 12π/7 - 2π = -2π/7. So, cos(8π/7) = cos(-6π/7) = cos(6π/7), and cos(12π/7) = cos(-2π/7) = cos(2π/7). Therefore, cos(4π/7) + cos(8π/7) + cos(12π/7) = cos(4π/7) + cos(6π/7) + cos(2π/7) = same as the previous sum, which is -1/2. [Continue]
Therefore, both sums are equal to -1/2. [Continue]
So, substituting back into Sum:\n\nSum = 18 - 8*(-1/2) + 2*(-1/2) = 18 + 4 - 1 = 21. [Continue]
Therefore, the sum over k=1,2,3 of (2*sin(kπ/7))^4 is 21. [Continue]
But wait, that's just the sum for each k=1,2,3. But in our problem, each of these lengths occurs 7 times. So, the total sum S is 7*(21) = 147. [Continue]
Wait, hold on. Let me double-check that.
Let's keep only the essential derivation, avoid repetition, and proceed efficiently. So, the sum over k=1,2,3 of (2*sin(kπ/7))^4 is 21, as we just found. Since each chord length occurs 7 times, the total sum is 7*21 = 147. [Continue]
Therefore, the sum S is 147. Then, we need to find 147 mod 100, which is 47.
The reasoning is sufficient; state only the final conclusion briefly and close the reasoning now. So, the answer is 47.\n\n**Final Answer**\n\\boxed{47}\n</think>
To find the sum of the 4th powers of the lengths of all 21 edges and diagonals of a regular 7-gon inscribed in a unit circle, we start by noting that the lengths of the edges and diagonals correspond to chords of the circle subtended by angles of
[... omitted]
Finally, finding the remainder when 147 is divided by 100:\n\n\\[\n147 \\mod 100 = 47\n\\]\n\nThe final answer is:\n\n\\[\n\\boxed{47}\n\\]

Correct and Efficient Reasoning

Figure 20: MetaCtrl reasoning on the same AMC example. By dynamically regulating how the frozen reasoner proceeds, MetaCtrl follows a more direct derivation and avoids prolonged unproductive deliberation, reaching the correct answer with 2,479 reasoning tokens.

Case Study 7: GPQA

Question: The reaction of an electron pair donor, nucleophile (Nu) with an electron pair acceptor is called nucleophilic substitution reaction. An sp3-hybridized electrophile needs to have a leaving group to proceed with
[... omitted]
Hydroxide 3. Propionate 4. Methanol 5. Ethanethiolate\n\n(A) 5, 2, 1, 3 and 4\n(B) 2, 5, 1, 4 and 3\n(C) 5, 2, 3, 1 and 4\n(D) 2, 5, 3, 4 and 3\n\nPlease write your final answer in the form of \\boxed{A}, \\boxed{B}, \\boxed{C}, or \\boxed{D}
Answer: A

Vanilla Reasoning Total Token Number: 12806

Okay, so I need to arrange these nucleophiles in order of their reactivity in an aqueous solution, from the most reactive to the
[... omitted]
First, I remember that in polar protic solvents like water, nucleophilicity is influenced by the ability of the nucleophile to donate electrons and how well it can be solvated. Stronger nucleophiles are those that are less solvated, meaning they are less able to be surrounded by solvent molecules, which can hinder their ability to attack the electrophilic center.
[... omitted]
4-methylcyclohexan-1-olate: This is the conjugate base of 4-methylcyclohexanol. So it's a carbanion, right? Because it's the deprotonated form of an alcohol. So it's a strong nucleophile, but in aqueous solution, carbanions are usually not very stable and might be less reactive because they are more basic and can be protonated.
[... omitted]
Wait, if the correct order is 3 > 5 > 2 > 1 > 4, then none of the options match. But maybe I made a mistake in the basicity of the carbanion. Let me think about the carbanion again.
The carbanion (1) is the conjugate base of 4-methylcyclohexanol, which has a pKa of ~17-18. Hydroxide is the conjugate base of water, pKa ~15.7. So, the carbanion is a stronger base than hydroxide. Therefore, in aqueous solution, the carbanion is more solvated than hydroxide, making it less nucleophilic. So, hydroxide is more nucleophilic than the carbanion. But earlier I thought that the carbanion is more nucleophilic than hydroxide, but that might be incorrect.
Wait, this is a critical point. If the carbanion is a stronger base than hydroxide, it is more likely to be solvated by water, making it less nucleophilic. So, hydroxide is more nucleophilic than the carbanion.
[... omitted]
Wait, option (C) is 5, 2, 3, 1 and 4 → 5 > 2 > 3 > 1 > 4
If the order is 5 > 2 > 3 > 1 > 4, that would mean ethanethiolate (5) is more nucleophilic than hydroxide (2), which is more nucleophilic than propionate (3), which is more nucleophilic than carbanion (1), which is more nucleophilic than methanol (4). But according to my earlier reasoning, propionate (3) is more nucleophilic than ethanethiolate (5). So this is conflicting.
Wait, maybe I made a mistake in comparing propionate and ethanethiolate. Let me think about their nucleophilicity. Propionate is a carboxylate, which is resonance-stabilized. Ethanethiolate is a thiolate, which is less resonance-stabilized.
[... omitted]
Ethanethiolate (5) is a larger ion than hydroxide (2), so it's less solvated, making it more nucleophilic than hydroxide.
Propionate (3) is a carboxylate, which is resonance-stabilized, but in aqueous solution, it's less nucleophilic than thiolates because of the size and the ability to donate electrons.
Hydroxide (2) is a good nucleophile, but less than thiolates.
Carbanion (1) is a very strong nucleophile, but in aqueous solution, it might be less nucleophilic than thiolates because of solvation. Wait, but carbanions are more basic and more likely to be protonated by water, making them less reactive.
Methanol (4) is a weak nucleophile.
So, the order would be: ethanethiolate (5) > hydroxide (2) > propionate (3) > carbanion (1) > methanol (4). But this is not matching any options either.
[... omitted]
Wait, but this is a general order. For example, sulfide ions (S^{2-}) are more nucleophilic than hydroxide in aqueous solution. Similarly, thiolates (RS^-) are more nucleophilic than carboxylates ($RCOO^-$). So, in this case, ethanethiolate (5) is more nucleophilic than propionate (3). Hydroxide (2) is more nucleophilic than propionate (3). So, the order would be 5 > 2 > 3 > 1 > 4. This matches option (C): 5, 2, 3, 1, 4.
Wait, but what about the carbanion (1)? If it's a carbanion, it's more nucleophilic than hydroxide? Or less? In aqueous solution, carbanions are more likely to be protonated by water, making them less nucleophilic. So, if the carbanion is more basic, it's more likely to be protonated, making it less nucleophilic. So, in this case, the carbanion (1) would be less nucleophilic than hydroxide (2). So, the order would be 5 > 2 > 3 > 1 > 4, which is option (C).
But earlier I thought that the carbanion is more nucleophilic than hydroxide, but that might not be the case in aqueous solution.
[... omitted]
Therefore, the answer is option (C).\n\n**Final Answer**\n\\boxed{C}\n</think>
To determine the reactivity of the given nucleophiles in **aqueous solution**, we must consider the **nucleophilicity**
[... omitted]
(3)** – more nucleophilic than carbanion but less than hydroxide\n4. **4-methylcyclohexan-1-olate (1)** – least nucleophilic due to high solvation\n5. **Methanol (4)** – least reactive (neutral, weak nucleophile)\n\n---\n\n### Final Answer\n\n$$\n\\boxed{C}\n$$

✕ **Wrong Answer**

Figure 21: Vanilla reasoning on a GPQA chemistry example. The model repeatedly revisits conflicting hypotheses about solvation, basicity, and nucleophilicity, and an early mischaracterization of the alkoxide contributes to a prolonged inconsistent trajectory, ultimately yielding an incorrect answer after 12,806 reasoning tokens.

Case Study 7: GPQA

Intervene Action: Continue → Fast Think → Skip Think → Continue → Continue → Continue → Continue → Skip Think → Continue → Stop Think

MetaCtrl Reasoning Total Token Number: 2001

Okay, so I need to arrange these nucleophiles in order of their reactivity in an aqueous solution, from the most reactive to the least. The options are 1 to 5, and the choices are A to D. Let me start by recalling what factors affect nucleophilicity in aqueous solutions. [Continue]
First, I remember that in polar protic solvents like water, nucleophilicity is influenced by the ability of the nucleophile to donate electrons and how well it can be solvated. Stronger nucleophiles are those that are less solvated, meaning they are less able to be surrounded by solvent molecules, which can hinder their ability to attack the electrophilic center. So, in aqueous solutions, the nucleophilicity order can be different from what you might expect in aprotic solvents.
Let's keep only the essential derivation, avoid repetition, and proceed efficiently. Let me list the nucleophiles again:\n\n1. 4-methylcyclohexan-1-olate (which is the conjugate base of 4-methylcyclohexanol)\n2. Hydroxide (OH^-)\n3. Propionate (which is the conjugate base of propionic acid, so propionate is $CH2CH2COO^-$)\n4. Methanol (CH3OH)\n5. Ethanethiolate ($CH2CH2S^-$)\n\nWait, but wait. Wait, the question says "nucleophiles" but some of these are neutral molecules. Wait, but in aqueous solution, they might be deprotonated. Wait, but the question says "nucleophiles" so maybe they are all in their deprotonated forms? Or are they as written?
Let's solve this directly with the minimum necessary reasoning and move to the answer. Let me think about each one's basicity and nucleophilicity. [Continue]
In aqueous solution, the nucleophilicity of a species is often related to its basicity. However, in polar protic solvents, the nucleophilicity can be influenced by the solvent's ability to solvate the nucleophile. For example, larger, more polarizable species may be less solvated, hence more nucleophilic. Also, the ability to donate electrons (basicity) is important. [Continue]
But wait, in aqueous solution, the nucleophilicity order for common nucleophiles is often: I^- > Br^- > Cl^- > H2O > RO^- > NH3 > HO^- > ... Wait, but that's in aprotic solvents. Wait, no, in polar protic solvents, the order is different. For example, in water, the nucleophilicity of anions is influenced by their size and polarizability. Larger anions are more polarizable and hence more nucleophilic. But also, the basicity is related to the pKa of their conjugate acids. For example, a stronger base is a stronger nucleophile in polar protic solvents, but sometimes the opposite can happen due to solvation effects. [Continue]
Wait, but here we have different species. Let me think about each one:\n\n1. 4-methylcyclohexan-1-olate: This is the conjugate base of 4-methylcyclohexanol. The conjugate acid is a tertiary alcohol. The pKa of alcohols is around 16-18. So the
[... omitted]
their nucleophilicity is lower. So, larger anions are less solvated, hence more nucleophilic. So, for example, in water, the order of nucleophilicity for similar anions would be: I^- > Br^- > Cl^- > OH^- > RO^-. But here, we have different anions. [Continue]
Wait, but let's compare the given nucleophiles. Let me list them again:\n\n1. 4-methylcyclohexan-1-olate (let's call it RO^-, where R is a bulky group)\n2. Hydroxide (OH^-)\n3. Propionate ($CH2CH2COO^-$, which is the conjugate base of propionic acid,
[... omitted]
Wait, but the question lists them as 1 to 5. Let me check the original question again.
Let's solve this directly with the minimum necessary reasoning and move to the answer. Let me think about each one's basicity and nucleophilicity. [Continue]
In aqueous solution, the nucleophilicity of a species is often related to its basicity. However, in polar protic solvents, the nucleophilicity can be influenced by the solvent's ability to solvate the nucleophile. For example, larger, more polarizable
[... omitted]
overall order is \(5 > 2 > 1 > 3 > 4\), which corresponds to option A.
The reasoning is sufficient; state only the final conclusion briefly and close the reasoning now. The correct order is 5 (ethanethiolate) > 2 (hydroxide) > 1 (cyclohexanolate) > 3 (propionate) > 4 (methanol). So the order from most to least reactive is 5, 2, 1, 3, 4. Which corresponds to option A: 5, 2, 1, 3, 4. So the answer is \boxed{A}.\n</think>
To determine the reactivity of the given nucleophiles in an **aqueous solution**, we must consider the **nucleophilicity** of each species in this type of solvent. Aqueous solutions are **polar protic**, meaning they contain water molecules that
[... omitted]
Least Reactive):\n\n**5 (Ethanethiolate) > 2 (Hydroxide) > 1 (Cyclohexanolate) > 3 (Propionate) > 4 (Methanol)**\n\nThis corresponds to **Option A**.\n\n---\n\n### Final Answer:\n\n$$\n\\boxed{A}\n$$

Correct and Efficient Reasoning

Figure 22: MetaCtrl reasoning on the same GPQA example. By dynamically regulating how the frozen reasoner proceeds, MetaCtrl substantially reduces prolonged hypothesis exploration and reaches the correct answer with 2,001 reasoning tokens.

Case Study 8: GPQA

Question: Consider an aperture, which shapes like a N-sided polygon. All the apothems are of the same length a. When a monochromatic light (of wavelength \\lambda), travelling along z-axis, falls vertically on this aperture, it creates a series of intensity maxima and minima in the far field region. What is the angular distance between the first two minima when N is infinitely large ? Assume that the related angles are small so that tan theta = theta approximation can be taken.\n\n(A) 0.506 \\lambda / a\n(B) 1.220 \\lambda /a\n(C) 0.500 \\lambda / a\n(D) 0.610 \\lambda / a\n\nPlease write your final answer in the form of \\boxed{A}, \\boxed{B}, \\boxed{C}, or \\boxed{D}
Answer: A

Vanilla Reasoning Total Token Number: 3702

Okay, so I need to figure out the angular distance between the first two minima when the aperture is an N-sided polygon with all apothems of length a, and N is infinitely large. The light is monochromatic with wavelength λ, traveling along the z-axis. The angles are small, so we can use the small angle approximation where tanθ ≈ θ. The options are given, so maybe I can
[... omitted]
For a circular aperture, the first minimum is given by the formula θ ≈ 1.22 λ / (2a) = 1.22 λ / (2a) = 0.61 λ / a? Wait, but the first minimum for a circular aperture is at 1.22 λ / D, where D is the diameter. If the diameter is 2a, then θ ≈ 1.22 λ / (2a) = 0.61 λ / a. But the question is about the angular distance between the first two minima. Wait, but for a circular aperture,
[... omitted]
diameter 2a. Therefore, the first minimum for a circular aperture is at θ ≈ 1.22 λ / D = 1.22 λ / (2a) = 0.61 λ /a. Therefore, the angular distance between the first two minima (which are symmetric around the central maximum) would be 2*0.61 λ /a = 1.22 λ /a. Therefore, the answer would be B. But let me check the options again. Option B is 1.22 λ /a. So that would be the answer? But wait, the problem says \"the angular distance between the first two minima\". So if the first minima are at ±θ, then the angular distance between them is 2θ. So if θ is 1.22 λ / (2a), then 2θ is 1.22 λ /a. So that would be option B. But let me check if there's another approach.\n\nAlternatively, maybe the problem is not about a circular aperture. Wait, but if N is
[... omitted]
diffraction pattern of a polygon with N sides, but when N is infinite, the number of sides is infinite, so it's a circle. Therefore, the answer is B. However, let me check the options again. Option B is 1.22 λ /a. But wait, in the case of a circular aperture, the first minimum is at 1.22 λ / D, where D is the diameter. If D is 2a, then 1.22 λ / (2a) = 0.61 λ /a. So the angular distance between the first two minima is 2*0.61 λ /a = 1.22 λ /a. Therefore, answer is B. But let me check if the question is about the angular distance between the first two minima, which would be the distance between the first minima on either side of the central maximum. So that would be 2θ, where θ is the angle of the first minimum. Therefore, if θ is 1.22 λ / D, then 2θ is 1.22 λ / D * 2? Wait, no. Wait, θ is 1.22 λ / D. So the angular distance between the two minima is 2θ. Wait, no. Wait, if the first minimum is at angle θ from the center, then the angular distance between the two minima is 2θ. So if θ is 1.22 λ / D, then the angular distance is 2*(1.22 λ / D). But D is 2a, so 2*(1.22 λ / (2a)) = 1.22 λ /a. Therefore, answer is B. So that would be option B. But let me check if there's another way to think about this.\n\nAlternatively, maybe the problem is not about a circular
[... omitted]
B is 1.22 λ /a. But let me think about the single-slit diffraction. For a single slit of width a, the first minimum is at θ = λ / a. But here, if the aperture is a circle with diameter 2a, then the first minimum is at 1.22 λ / (2a) = 0.61 λ /a. Therefore, the angular distance between the first two minima is 2*0.61 λ /a = 1.22 λ /a. Therefore, answer is B. So I think that's the answer. But let me check if there's another approach.\n\nAlternatively, maybe the problem is about the diffraction pattern of a polygon with N sides, and when N is infinite, the number of sides is infinite, so it's a circle. However, maybe the apothem is different. Wait,
[... omitted]
the angular distance between the first two minima is indeed 1.22 λ /a. Let me check the formula for the circular aperture. The first minimum occurs at θ ≈ 1.22 λ / D, where D is the diameter. Therefore, the angular distance between the first minima on either side is 2θ, which would be 2*(1.22 λ / D). If D is 2a, then 2*(1.22 λ / (2a)) = 1.22 λ /a. Therefore, yes, that's correct. Therefore, answer is B. However, let me check if there's a different approach where the answer is different. For example, if the aperture is considered as a polygon with N sides, and when N approaches infinity, maybe the diffraction pattern is similar to a single slit? But no, because a polygon with N sides would have multiple slits? Wait, no. Wait, the aperture is a single
[... omitted]
between the two minima is 1.22 λ /a. Therefore, answer is B. Therefore, I think the answer is B. However, let me check if there's a different approach where the answer is different. For example, if the aperture is considered as a polygon with N sides, and when N approaches infinity, the diffraction pattern is similar to a circular aperture. Therefore, the answer is B. Therefore, I think the answer is B.\n\nBut wait, let me think again. The problem says \"the angular distance between the first two minima\". If the first minima are at angles θ and -θ, then the angular distance between them is 2θ. For a circular aperture, θ is 1.22 λ / D. If D is 2a, then θ is 1.22 λ / (2a) = 0.61 λ /a. Therefore, the angular distance between the two minima is 2*(0.61 λ /a) = 1.22 λ /a. Therefore, answer is B. Therefore, I think the answer is B. Therefore, I will go with option B.\n\n**Final Answer**\n\\boxed{B}\n</think>
To determine the angular distance between the first two minima in the far-field diffraction pattern of an N-sided polygonal aperture with all apothems of length $ a $, and as $ N \\to \\infty $, we must consider the limiting behavior of the
[... omitted]
angular distance between these two minima is:\n\n$$\n\\text{Angular distance} = 2\\theta = 2 \\times \\frac{0.61 \\lambda}{a} = \\frac{1.22 \\lambda}{a}\n$$\n\n---\n\n### Final Answer\n\n$$\n\\boxed{B}\n$$

✗ **Wrong Answer**

Figure 23: Vanilla reasoning on a GPQA physics example. The model repeatedly interprets the "first two minima" as the symmetric first minima on opposite sides of the central maximum, leading to prolonged verification of the same interpretation and ultimately an incorrect answer after 3,702 reasoning tokens.

Case Study 8: GPQA

Intervene Action: Continue → Fast Think → Continue → Skip Think → Continue → Continue → Fast Think → Continue → Continue → Stop Think

MetaCtrl Reasoning Total Token Number: 3117

Okay, so I need to figure out the angular distance between the first two minima when the aperture is an N-sided polygon with all apothems of length a, and N is infinitely large. The light is monochromatic with wavelength λ, traveling along the z-axis. The angles are small, so we can use the small angle approximation where tanθ ≈ θ. The options are given, so maybe I can recall some diffraction formulas. [Continue]

First, let me recall that for a polygonal aperture, when N is large, it might approximate a circular aperture? Wait, but the problem says N is infinitely large. Hmm. Wait, but the apothem is the same for all sides. The apothem of a regular polygon is the distance from the center to the midpoint of a side, which is also the radius of the inscribed circle. So if all apothems are the same length a, then as N approaches infinity, the polygon becomes a circle with radius a. Wait, but the apothem of a regular polygon with N sides is given by a = (s)/(2 tan(π/N)), where s is the side length. But as N approaches infinity, tan(π/N) ≈ π/N, so a ≈ s/(2*(π/N)) = s*N/(2π). Therefore, s ≈ 2πa/N. But as N approaches infinity, the side length s approaches zero, and the polygon becomes a circle with radius a. So the aperture becomes a circular aperture with radius a. Therefore, the problem reduces to finding the angular distance between the first two minima in the diffraction pattern of a circular aperture, which is the Airy disk pattern.

Let's keep only the essential derivation, avoid repetition, and proceed efficiently. For a circular aperture, the first minimum occurs at an angle θ where sinθ = 1.22λ/(2a). Wait, wait, the formula for the first minimum in the Airy disk is given by the first zero of the Bessel function, which is at sinθ = 1.22λ/(D), where D is the diameter of the aperture. Wait, but here the radius is a, so diameter is 2a. Therefore, sinθ = 1.22λ/(2a). But since the angles are small, sinθ ≈ θ. Therefore, θ ≈ 1.22λ/(2a) = 0.61λ/a. But the question is about the angular distance between the first two minima. Wait, but in the circular aperture, the first minimum is at θ ≈ 0.61λ/a, and the next minimum would be... Wait, actually, the first minimum is the first one, and the next one is the second minimum. Wait, but in the Airy pattern, the first minimum is the first dark ring, and the next minima are further out. However, the angular distance between the first two minima would be the difference between the first and second minima. Wait, but maybe the question is referring to the angular distance between the first minimum and the second minimum? Or maybe between the first two minima, meaning the first and the second? Let me check. [Continue]

Wait, the problem says \"the angular distance between the first two minima\". So if there are multiple minima, the first two would be the first and the second. But in the case of a circular aperture, the first minimum is at θ1 = 1.22λ/(2a) ≈ 0.61λ/a, and the second minimum is at θ2 = 2.23λ/(2a) ≈ 1.115λ/a. Therefore, the angular distance between them would be θ2 - θ1 = (2.23 - 1.22)λ/(2a) = 1.01λ/(2a) ≈ 0.505λ/a. Wait, but that's approximately 0.505λ/a, which is close to option C (0.500λ/a) or A (0.506λ/a). But let me check the exact values.

Let's solve this directly with the minimum necessary reasoning and move to the answer. For a circular aperture, the intensity pattern is given by the Airy disk. The angular position of the minima is given by the zeros of the Bessel function of the first kind of order 1, J1(k a sinθ) = 0, where k = 2π/λ. The first zero of J1 occurs at k a sinθ = 3.8317, so sinθ = 3.8317/(k a) = 3.8317/(2π/a) * λ/(2π) ??? Wait, let me do it step by step. [Continue]

The formula for the first minimum in the diffraction pattern of a circular aperture is given by:\n\nsinθ = 1.22 λ / D,\n\nwhere D is the diameter of the aperture. Since the apothem is a, and for a circle, the radius is a, so D = 2a. Therefore, sinθ = 1.22 λ / (2a) = 0.61 λ / a. So θ ≈ 0.61 λ / a (since θ is small, sinθ ≈ θ). [Continue]

But the question is about the angular distance between the first two minima. So the first minimum is at θ1 = 0.61 λ / a, and the second minimum is at θ2 = ?

Let's keep only the essential derivation, avoid repetition, and proceed efficiently. The zeros of the Bessel function J1(x) occur at x = 3.8317, 7.0156, 10.1735, etc. So the first minimum is at x = 3.8317, the second at x = 7.0156. Therefore, the angular positions are given by x = k a sinθ, where k = 2π/λ. Therefore:\n\nFor the first minimum: 3.8317 = (2π/λ) a sinθ1 => sinθ1 = 3.8317 λ / (2π a) ≈ 3.8317 / (6.2832) * λ / a ≈ 0.61 * λ / a, which matches the previous result. [Continue]

Similarly, for the second minimum: 7.0156 = (2π/λ) a sinθ2 => sinθ2 = 7.0156 λ / (2π a) ≈ 7.0156 / 6.2832 * λ / a ≈ 1.116 * λ / a. [Continue]

Therefore, the angular distance between the first and second minima is θ2 - θ1 ≈ (1.116 - 0.61) * λ / a = 0.506 * λ / a. Which is approximately 0.506 λ / a, which is option A. But wait, the options are:\n\n(A) 0.506 λ / a\n\n(B) 1.220 λ /a\n\n(C) 0.500 λ / a\n\n(D) 0.610 λ / a\n\nSo according to this, the answer would be A. But wait, let me check again. Wait, the angular distance between the first two minima is θ2 - θ1. But if θ1 is the first minimum, and θ2 is the second, then the angular distance between them is θ2 - θ1. But in the problem statement, it says \"the angular distance between the first two minima\". So if the first minimum is at θ1 and the second at θ2, then the angular distance is θ2 - θ1. But according to the calculation, that is approximately 0.506 λ / a, which is option A. However, I need to check if the question is referring to the angular distance between the first minimum and the second minimum, or between the first and the second minima in some other sense. Wait, but the first two minima would be the first and the second. So the angular distance between them is θ2 - θ1. But let me check the exact value.

The reasoning is sufficient; state only the final conclusion briefly and close the reasoning now. The angular distance between the first two minima is approximately 0.506 λ / a, which corresponds to option A. However, I need to check if there's a

[... omitted]

distance between the first two minima is θ2 - θ1 ≈ 0.506λ/a. Therefore, the answer is A. However, I need to check if the question is referring to\n</think>

To determine the angular distance between the first two minima in the diffraction pattern of a polygonal aperture with an infinite number of sides (i.e., a circular aperture), we begin by recognizing that as $ N \\to \\infty $, the polygon becomes a

[... omitted]

\\approx \\frac{3.1839}{6.2832} \\cdot \\frac{\\lambda}{a} \\approx 0.506 \\cdot \\frac{\\lambda}{a}\n$$\n\n---\n\n### Final Answer\n\n$$\n\\boxed{A}\n$$

Correct and Efficient Reasoning

Figure 24: MetaCtrl reasoning on the same GPQA example. The controlled trajectory revisits the interpretation of the diffraction minima and uses the first two zeros of the Airy pattern, reaching the correct answer with 3,117 reasoning tokens.