Critical Positions in LLM Reasoning: A Metacognitive Lens on Intervention-Sensitive Locations

Wendi Cui, Anni Huang, Wei Wang, Damien Lopez, Sirui Bi
Intuit

Abstract

Large Language Models (LLMs) achieve strong reasoning performance by generating intermediate reasoning trajectories through autoregressive decoding. Existing approaches such as self-consistency, tree-based reasoning, and critical token analysis improve reasoning by exploring diverse trajectories or identifying influential tokens. In this work, we identify a distinct phenomenon in LLM reasoning: **critical positions** — locations where re-sampling continuations disproportionately influences the final outcome, independent of the specific token at that position. Motivated by this, we propose **Metacognitive Influence Tracing (`MIT`)**, the first unsupervised framework for identifying critical positions from a single reasoning trace using only attention structure. `MIT` models reasoning trajectories as influence graphs and estimates propagated influence through a diffusion process, enabling the discovery of structurally important reasoning positions without supervision or contrastive trajectories. Building on these insights, we further propose **Adaptive Critical Sampling (`ACS`)**, an intervention framework that selectively resamples at critical positions instead of regenerating entire reasoning traces. Across six benchmark tasks and five models, `ACS` improves by an average of **7.78 points** over CoT while reducing computational cost by **59.83%** over Self-Consistency.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable reasoning capabilities by generating intermediate reasoning trajectories. Prompting strategies such as Chain-of-Thought (CoT) (Wei et al., 2022) and reasoning-oriented models such as DeepSeek-R1 (Guo et al., 2025) leverage this paradigm by giving models additional "time to think" before producing final answers. However, reasoning trajectories remain highly fragile: small early mistakes can propagate through subsequent steps and ultimately lead to confidently incorrect conclusions (Besta et al., 2024; Yao et al., 2023; Becker and Soatto, 2024).

To improve reasoning robustness, prior work has largely focused on increasing reasoning diversity or validating generated trajectories. **Self-Consistency** (Wang et al., 2023) samples multiple reasoning trajectories and aggregates their answers, while **tree-based reasoning methods** (Yao et al., 2023; Hao et al., 2023) explicitly explore diverse reasoning paths. Other approaches introduce **verification** or **correction mechanisms** to detect and revise erroneous reasoning steps (Cobbe et al., 2021; Lightman et al., 2024; Li et al., 2023). Recently work on **critical tokens** (Lin et al., 2025; Bigelow et al., 2025) showed that certain tokens causally influence reasoning outcomes through their lexical content.

Despite their advantages, existing methods largely assume that all *positions* within a reasoning trajectory are equally important, which leads to either regenerating entire reasoning traces, or focus on the lexical identity of individual tokens, where changing the token leads to different outcomes. None of them identify whether certain *positions* in the reasoning process exert disproportionate influence on the final outcome regardless of whether the specific token is changed.

In this work, we identify a distinct phenomenon in LLM reasoning: **critical positions** — locations within a reasoning trajectory where re-sampling continuations disproportionately improves the final performance, independent of the specific token at that position. Unlike critical tokens, which are defined by lexical causality, critical positions arise from their *structural role* within the reasoning process. We show that these positions are both identifiable and actionable across models with explicit thinking and reasoning capabilities (e.g. DeepSeek-R1), which we denote *Reasoning Models*, and those without (e.g. Llama-3), which we denote *Non-reasoning Models* for the remaining of this paper.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 31357–31390
October 24-29, 2026 ©2026 Association for Computational Linguistics

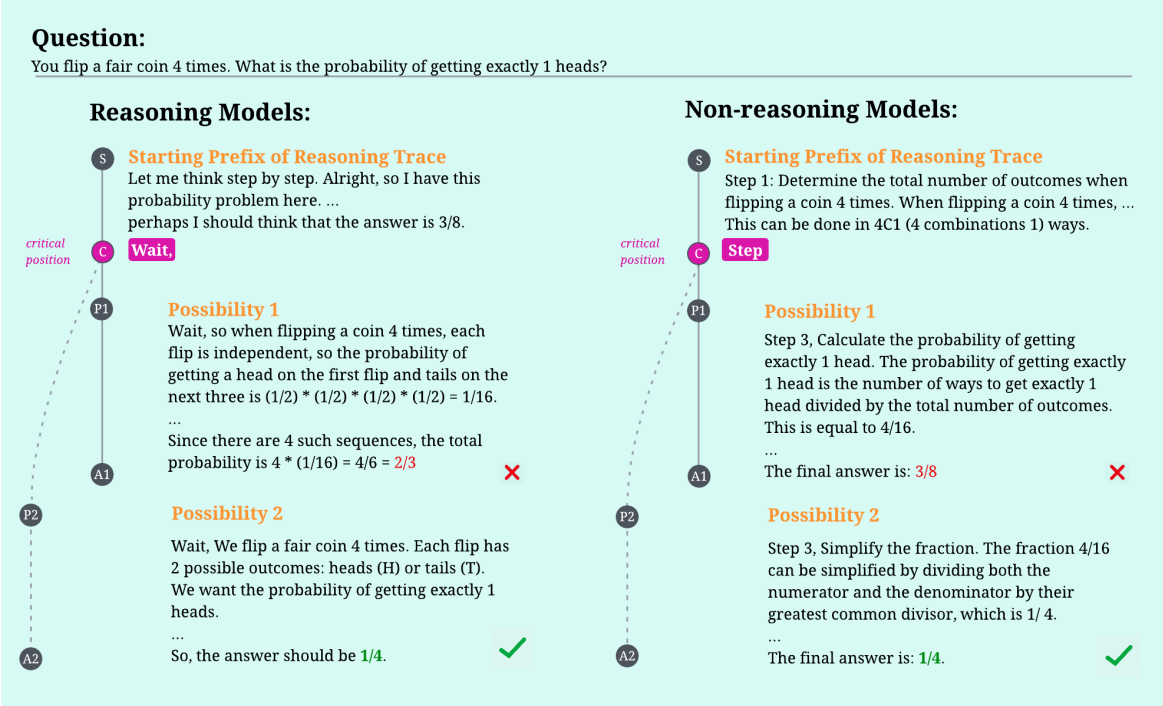


Figure 1: Illustration of **critical positions** in reasoning trajectories for both *Reasoning Models* (e.g., DeepSeek-R1) and *Non-reasoning Models* (e.g., LLaMA-3). Existing reasoning approaches typically treat all positions in a trajectory as equally important. In contrast, we identify specific positions where re-sampling continuations disproportionately influences the final outcome. In the example above, re-sampling from the critical position transforms an incorrect reasoning trajectory into a correct one, despite sharing the same reasoning prefix. We further observe that models exhibit recurring metacognitive markers around these positions, such as *"Wait"* and *"Step"*.

Motivated by this, we propose **Metacognitive Influence Tracing (MIT)**, the first unsupervised framework for identifying critical positions from a single reasoning trajectory using only attention structure. Rather than relying directly on raw attention scores, MIT adopts three **influence principles** inspired by cognitive science, and models reasoning trajectories as **influence graphs** where attention patterns form the initial connections. It then estimates propagated influence through a diffusion process based on the heat kernel. This enables MIT to uncover latent structural dependencies within reasoning traces without supervision, contrastive trajectories, or ground-truth labels.

Building on these insights, we further propose **Adaptive Critical Sampling (ACS)**, an inference-time intervention framework that selectively resamples at identified critical positions instead of regenerating entire reasoning trajectories. By concentrating computation on structurally influential positions, ACS improves reasoning reliability while substantially reducing redundant sampling. Across six reasoning benchmarks and five models, ACS achieves an average accuracy improvement of **7.78 points** while reducing computational cost by **59.83%** relative to self-consistency. Our contributions are summarized as follows:

- We introduce **critical positions**, a structural phenomenon in LLM reasoning trajectories distinct from prior notions of critical tokens.
- We propose **Metacognitive Influence Tracing (MIT)**, the first unsupervised framework for identifying critical positions from a single reasoning trajectory through a metacognitive lens using only attention structure.
- We propose **Adaptive Critical Sampling (ACS)**, an inference-time intervention framework that improves reasoning accuracy while substantially reducing the computational cost of self-consistency.

2 Critical Positions in LLM Reasoning

2.1 Definition and Hypothesis

We define a *critical position* in a reasoning trace as a location p whose correctness ratio — the fraction of rollout continuations from p that yield a correct answer — exceeds the correctness ratio of

Token–Answer Decoupling			Intervention Effectiveness (Δ Acc. vs. SC)		
Event	**Reasoning**	**Non-reasoning**	**Intervention Type**	**Reasoning**	**Non-reasoning**
Token changes on resample	86.5%	54.7%	Most critical position	**+11.11**	**+27.27**
Answer changes on resample	42.8%	27.1%	Random cognitive marker	-3.70	−6.82
Both token & answer change	35.0%	14.6%	Random position	−11.11	−9.09
Token changes, answer stays	51.5%	40.1%	No intervention (SC)	0	0

Table 1: Critical position analysis on GPQA-DIAMOND. **Left panel:** token–answer decoupling at critical positions. Token changes are frequent but do not drive answer changes, confirming that critical positions are structural, not lexical. **Right panel:** accuracy gain over SC baseline by intervention type. Resampling at critical positions delivers large gains while resampling at random cognitive marker or random positions yields negative deltas.

self-consistency at the same rollout budget by at least τ, a threshold hyperparameter. This effect must be robust to variation in the specific token generated at p. The *most critical position* is the argmax over correctness ratio among all critical positions. This distinguishes critical positions from critical tokens (Lin et al., 2025), which are defined by lexical causality — replacing the token directly changes the outcome. Critical positions are characterized by their *structural role* in the reasoning trace, not by their lexical content. We hypothesize that critical positions are a fundamental property of LLM reasoning traces, identifiable across model families, reasoning paradigms, and task types.

2.2 Critical Positions Are in Every Trace

To empirically validate that critical positions exist as a real phenomenon — independent of any particular identification method — we conduct a rollout-based study. For each token position within an incorrect trace, we generate 64 rollout continuations and compute the correctness ratio of the resulting answers. We collect 50 incorrect reasoning traces from the task GPQA-DIAMOND for both reasoning model `DeepSeek-Distilled-Qwen-1.5B`, and non-reasoning model `LLaMA-3.1-8B`, and are able to identify critical positions in all 100 entries with $\tau = 10\%$. Ablation of this choice is in Appendix A.4.

2.3 Critical Positions are Token-Decoupled

To understand whether critical positions are the same as critical tokens — where changing the token directly alters the outcome — we measure the correlation between token changes at a critical position and changes in the final answer. As shown in Table 1 (left panel), in **51.5%** of reasoning-model resamples and **40.1%** of non-reasoning resamples, the token at the critical position changes but the final answer does not. This means that in the majority of cases, lexical substitution at the critical position leaves the outcome unchanged — the token is not the causal agent. Conversely, only 35.0% (reasoning) and 14.6% (non-reasoning) of resamples produce a change in both token and answer, confirming that co-occurrence of token and answer change is the exception, not the rule.

This result distinguishes critical positions from critical tokens: what matters at these positions is not the specific token generated, but the position's structural role in the reasoning trace. We interpret critical positions as *markers of intervention sensitivity* — points in the reasoning process where resampling tends to be most impactful for downstream outcomes.

2.4 Critical Positions Manifest as Cognitive Markers

Our analysis further reveals that critical positions are not uniformly distributed throughout reasoning trajectories. Instead, tokens appearing at these positions exhibit recurring linguistic patterns, which we denote as **cognitive markers**. These markers frequently emerge at structural transition points in the reasoning process, such as moments of reconsideration, decomposition, or procedural planning.

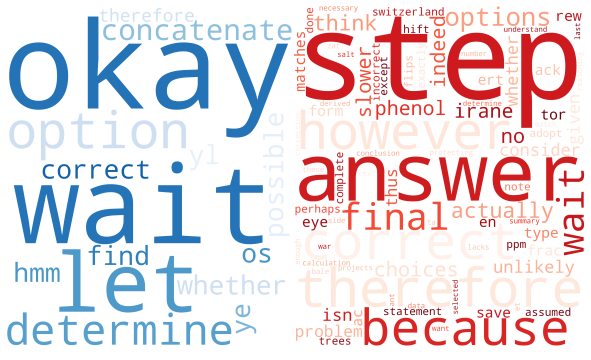


Figure 2: Distribution of tokens at the critical positions. Reasoning models (left) concentrate heavily on *'wait'* and *'okay'*, while non-reasoning models (right) show more diverse procedural markers.

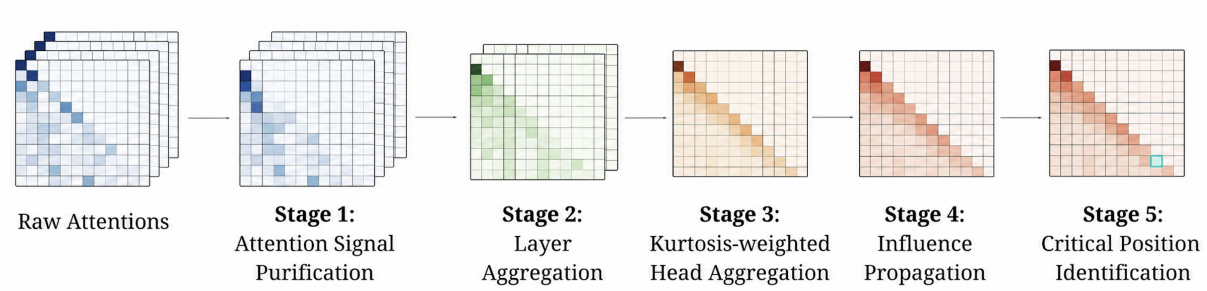


Figure 3: The MIT algorithm, surfacing a single critical position from raw multi-layer, multi-head attention matrices through cleaning, aggregations, propagation, and scoring.

As illustrated in Figure 2, reasoning models exhibit highly concentrated marker usage: *"okay"* and *"wait"* together account for **79.6%** of all critical positions. This strong convergence suggests that reasoning models have developed consistent internal deliberation mechanisms, using these markers to regulate cognitive state transitions before influential reasoning steps.

In contrast, non-reasoning models display a broader distribution of procedural markers, including *"step"* (23.2%), *"answer"* (9.4%), and logical connectives such as *"therefore"* (3.3%). These markers reflect a reasoning strategy that relies more heavily on explicit textual organization and sequential decomposition.

This aligns with Qian et al. (2025), who find sharp mutual information peaks between hidden states and the correct answer at exactly these token types, providing validation that cognitive markers co-occur with positions of structural significance.

2.5 Markers as Indicators, Not Causes

The findings in §2.3 and §2.4 create an apparent tension: if critical positions are token-decoupled, why do specific tokens — *"wait"*, *"okay"*, *"step"* — reliably occupy them? If token identity is non-causal, what explains their concentration at critical positions? Could the marker token itself be causing the effect?

To answer this, we run a controlled resampling experiment on the incorrect GPQA-DIAMOND traces. For each trace, we generate 8 continuation samples from a single chosen position and record the resulting accuracy. We compare three conditions for how the position is chosen:

- the **critical position** identified from the rollout
- a **randomly selected cognitive marker position** — a position chosen uniformly at random from all positions in the trace whose token is a cognitive marker (e.g., *"wait"*, *"okay"*, *"step"*)
- a **uniformly random position** with no constraint on token identity.

As shown in Table 1 (right panel), the results directly rule out token causality. Resampling at the most critical position yields substantial gains over SC (+11.11 pp for reasoning models, +27.27 pp for non-reasoning models). Resampling at a randomly selected cognitive marker position, produces negative deltas (−3.70 pp and −6.82 pp respectively), with similar pattern observed with resampling at a uniformly random position (−11.11 pp and −9.09 pp). The marker token, separated from the structural position, confers no benefit.

The resolution is that cognitive markers are **indicators**, not **causes**. Unlike Lin et al. (2025)'s critical tokens — where replacing the token changes the outcome — the marker itself carries no causal power: the same token at a non-critical position produces no gain. What makes a position critical is structural, not lexical; the marker is a correlate, not a constituent, of criticality.

3 Metacognitive Influence Tracing (MIT)

3.1 Intuition

To identify critical positions without exhaustive rollout, we adopt a metacognitive lens: rather than sampling from a position to observe its effect, we investigate how the model thinks and how the step-by step reasoning lead to the final answer to understand which position is critical.

Since positions are presented by tokens, identifying which position most influences the final answer is intrinsically a question about *inter-token relationships* — how strongly each earlier token influences the generation of the answer tokens. The attention mechanism is the only architectural component in a Transformer explicitly designed to model such relationships, making it the most direct internal signal available. This choice is further motivated by the convergence observed in §2.4: critical

positions cluster at cognitive markers, and Qian et al. (2025) independently show that these same token types correspond to sharp mutual information peaks between hidden states and the correct answer. Together, this suggests that the model's internal representations at cognitive marker positions are qualitatively distinct, making attention read from the answer tokens back through the trace a natural proxy for locating critical positions.

However, raw attention scores suffer from known artifacts like attention sinks (Xiao et al., 2024) and over-attention to punctuation (Clark et al., 2019), where disproportionate attention on a few tokens obscures true logical flows. Additionally, raw attention captures only *direct* influence, missing multi-step dependencies that are essential in extended reasoning traces. To overcome these limitations, `MIT` introduces two core innovations: a set of cognitively motivated **Influence Principles** to purify the raw attention signal, and a **Propagation Model** based on network science to capture the multi-step nature of reasoning.

3.2 Influence Principles for Signal Purification

To isolate the true logical signal from the noise inherent in raw attention, we establish three principles. Inspired by cognitive science, we treat semantic tokens as proxies for **concepts** — a term widely used in cognitive science to denote the basic units of thought. These principles allow us to reinterpret the attention mechanism through a conceptual lens.

- **Influence is Contextual, Not Self-Referential.** A concept's meaning is defined by its interaction with other concepts. Inspired by the contextual modulation of neural responses in the brain (Reynolds and Desimone, 2003), we interpret the attention between different tokens as the primary signal of conceptual interplay. To focus on this relational information, our method therefore excludes a token's self-attention, which is a non-contextual operation.
- **Influence is Semantic, Not Syntactic.** Logical reasoning operates on the meaning of concepts, not their grammatical arrangement. Supported by psycholinguistic evidence that humans recall meaning (semantics) over form (syntax) (Sachs, 1967), we aim to analyze the conceptual backbone of the reasoning trace. To achieve this, our method filters out non-semantic tokens (e.g., punctuation, stopwords) that serve primarily grammatical rather than logical functions.
- **Influence Propagates.** The impact of a single concept is not confined to an immediate step but propagates through a chain. This is analogous to *spreading activation* in cognitive networks, where activating one concept sends a ripple of influence to others (Collins and Loftus, 1975). To capture this phenomenon, our framework moves beyond single-step attention scores to model the full, multi-step influence flow.

3.3 Influence Propagation as a Diffusion Process

While the first two influence principles are used to *purify* the initial influences from raw attention scores, the third principle focuses on the *propagation* of influences. To achieve that, we model the reasoning trace as a directed graph of tokens as nodes, and propose that the total, propagated influence can be measured by modeling a **diffusion process** on this graph. This approach naturally includes all possible influence paths while allowing us to control the extent of the diffusion.

This process is captured by the **matrix exponential**, which we augment with a damping factor γ to control the intensity of the influence spread. For a graph with an adjacency matrix A (our purified attention matrix), the total influence S is given by the Taylor series expansion of $\exp(\gamma A)$. The term $\frac{(\gamma A)^k}{k!}$ represents the influence over paths of length k, with the factorial denominator acting as a strong penalty for longer paths and γ scaling the overall propagation effect. The total influence is the sum over all possible path lengths:

$$S = \exp(\gamma A) = I + \gamma A + \frac{(\gamma A)^2}{2!} + \dots \quad (1)$$

where S is the final influence matrix. The damping factor γ is a tunable hyperparameter that controls the "reach" of influence; a smaller γ restricts influence to more local paths, while a larger γ allows more wide diffusion. This formulation allows us to compute the propagated influence in a single step.

3.4 The MIT Algorithm

We now describe the full **Metacognitive Influence Tracing (`MIT`)** pipeline, which systematically refines raw, noisy attention patterns from a Transformer to surface a single identified critical position for the final generated answer in a reasoning trace. The full procedure is in Appendix Algorithm 2.

3.4.1 Stage 1: Attention Signal Purification

The first stage applies our influence principles to the raw attention signal. For a given reasoning trace with K tokens, we start with the raw attention matrices, $\{A_{l,h} \in \mathbb{R}^{K \times K}\}$ for all layers l and heads h. We apply the following transformations to each raw attention matrix:

- **Contextual Filtering:** We enforce the principle of contextual influence by zeroing out the diagonal of each matrix, thereby removing self-attention scores.
- **Semantic Filtering:** We follow the semantic principle by zeroing out attentions to non-semantic tokens (e.g., stopwords, punctuation) and special tokens (e.g., <bos>).

The output of this stage is a set of purified attention matrices, $\{A'_{l,h} \in \mathbb{R}^{K \times K}\}$.

3.4.2 Stage 2: Layer Aggregation

To handle the layer dimension, we compute an average attention matrix for each head across all L layers. This creates one consensus matrix for each of the H heads, resulting in a tensor $A'' \in \mathbb{R}^{H \times K \times K}$.

$$A''_h = \frac{1}{L} \sum_{l=1}^{L} A'_{l,h} \quad \text{for each head } h \in [1, H] \quad (2)$$

3.4.3 Stage 3: Head Aggregation

This stage distills the multi-head tensor A'' into a single base influence matrix $A \in \mathbb{R}^{K \times K}$. We call the token doing the attending the *target token*, and the token being attended to the *source token*. Source tokens are strictly before the target token. We use kurtosis to compute head weights for each target token, hypothesizing that heads with more focused attention patterns are more involved in the reasoning for that specific step.

For each target token i, we calculate an importance weight, $w_{i,h}$, for each head h by computing the kurtosis of that head's attention distribution over all source tokens. A high kurtosis indicates a highly peaked, specialized attention pattern.

$$w_{i,h} = \text{Kurtosis}(A''_{h,i,:}) \quad (3)$$

These weights are then normalized across the heads to form $w'_{i,h}$. The final entry A_{ij} in our base influence matrix is the weighted sum of the influence scores from token j to token i across all heads:

$$A_{ij} = \sum_{h=1}^{H} w'_{i,h} \cdot A''_{h,ij} \quad (4)$$

The resulting matrix A represents the purified, consensus-based, one-step inter-token influences.

3.4.4 Stage 4: Influence Propagation

With the single, consensus-based influence matrix A, we now apply the principle of propagation by computing the matrix exponential defined in equation 1. This computation transforms the one-step influence matrix A into the final propagated influence matrix S, where each entry S_{ij} represents the total, accumulated influence of token j on token i over all possible paths.

3.4.5 Stage 5: Critical Position Identification

The final stage uses the propagated influence matrix S to find the single most influential position. Our goal is to identify which token j within the set of CoT steps (T_{CoT}) had the greatest total influence on the generation of the set of final answer tokens (T_{ans}). We calculate an influence score for each candidate token by summing its influence on all tokens in the final answer:

$$\text{Score}(j) = \sum_{i \in T_{ans}} S_{ij} \quad \text{for } j \in T_{CoT} \quad (5)$$

The token with the maximum score is designated as the critical position of the reasoning chain:

$$j_{critical} = \underset{j \in T_{CoT}}{\text{argmax}}\, \text{Score}(j) \quad (6)$$

This position, $j_{critical}$, is the final output of MIT.

4 Adaptive Critical Sampling

We introduce **Adaptive Critical Sampling (ACS)**, a framework that translates these metacognitive insights into significant efficiency and performance gains. Instead of brute-force, full-trace resampling, ACS performs a surgical intervention, focusing computational effort only on the most critical position identified by MIT. This allows for a more intelligent allocation of resources. The full procedure is detailed in Algorithm 1.

Algorithm 1 Adaptive Critical Sampling (ACS)

Require: Query Q, Model M, Number of samples N
1: Generate initial trace $T_0 \leftarrow M(Q)$
2: Identify critical position token $j_{critical} \leftarrow \text{MIT}(T_0)$
3: Let T_{prefix} be the tokens in T_0 up to $j_{critical}$
4: **for** $k = 1$ to N **do**
5: Generate new suffix $T_{suffix,k} \leftarrow M(T_{prefix})$
6: Store final answer from $T_{prefix} \circ T_{suffix,k}$
7: **end for**
8: **return** Majority vote of the stored final answers

Model	Method	Logical	Arithmetic		Commonsense	Symbolic		Avg.
		GPQA-D	MATH500	AIME24	StrategyQA	CoinFlip	LastLetter	
Reasoning models								
DeepSeek -Distilled -Qwen-1.5B	CoT	29.62	86.53	33.33	69.05	95.31	79.25	65.52
	CoT-SC	40.74	88.46	33.33	**83.33**	95.31	79.25	70.07
	ACS (Ours)	**44.44**	**92.31**	**33.33**	80.95	**98.43**	**94.34**	**73.97**
SmolLM3-3B (Thinking Mode)	CoT	53.90	97.62	44.83	61.86	100.00	100.00	76.37
	CoT-SC	57.45	100.00	48.28	76.29	100.00	100.00	80.34
	ACS (Ours)	**59.57**	**100.00**	**51.72**	**90.72**	**100.00**	**100.00**	**83.67**
Non-reasoning models								
LLaMA-3.1-8B	CoT	43.18	83.56	38.09	72.53	97.00	84.44	69.80
	CoT-SC	38.64	86.30	42.86	76.92	97.00	91.11	72.14
	ACS (Ours)	**61.36**	**89.04**	**57.14**	**80.22**	**99.00**	**93.33**	**80.02**
SmolLM3-3B (General Mode)	CoT	35.48	83.33	25.00	61.54	95.65	83.63	64.11
	CoT-SC	40.32	85.71	33.33	69.23	97.10	87.27	68.83
	ACS (Ours)	**46.77**	**85.71**	**33.33**	**76.92**	**98.55**	**89.09**	**71.73**
Gemma-3 -270M-IT	CoT	25.00	42.42	9.09	10.29	73.33	4.76	27.48
	CoT-SC	16.67	45.45	**27.27**	10.29	73.33	2.38	29.23
	ACS (Ours)	**33.33**	**48.48**	18.18	**16.18**	**73.33**	**7.14**	**32.77**

Table 2: Main results across six benchmarks and five models. We use accuracy percentage as the metric as multiple samples are generated. Our method ACS consistently outperforms CoT and CoT-SC across diverse reasoning tasks in both reasoning and non-reasoning models.

5 Experiments

5.1 Experimental Setup

Models. We evaluate on five different LLM configurations: `Llama-3.1-8B-Instruct`, `DeepSeek-R1-Distill-Qwen-1.5B`, `SmolLM3-3B` in both *general* and *thinking* modes, and `Gemma-3-270M-IT`. This includes models with explicit reasoning capabilities (Reasoning models) and those without (Non-reasoning models). We also intentionally focus on models from different families and sizes to showcase the broad applicability of our approaches.

Datasets. We group benchmarks by the primary reasoning skill they target: **Logical reasoning:** GPQA-Diamond (Rein et al., 2024); **Arithmetic reasoning:** MATH500 (Hendrycks et al., 2021) and AIME24 (Maxwell-Jia, 2024); **Commonsense reasoning:** StrategyQA (Geva et al., 2021); **Symbolic reasoning:** CoinFlip and Last Letter from (Wei et al., 2022).

Baselines and Hyperparameters. For baselines we compare with CoT and Self-Consistency with 9 samples, denoted CoT-SC. In the sampling process for ACS, we keep the prefix fixed and generate additional 8 candidates. This ensures that our approaches and Self-Consistency baseline have the same total number of samples. Temperature is set to be 0.7 throughout the experiments. For additional information in experiment set up please refer to Appendix Section A.2.2.

5.2 Main Results

Consistent Performance Gains. The primary finding from Table 2 is the consistent and significant performance improvement provided by ACS. Across all five model classes and all six reasoning benchmarks, ACS achieves the highest average accuracy. On average, ACS delivers a +7.78 point gain over CoT and a +4.31 point gain over CoT-SC. On the challenging GPQA-Diamond benchmark, ACS raises the accuracy of `LLaMA-3.1-8B` from 43.18% (CoT) and 38.64% (CoT-SC) to 61.36%, a jump of 18.18 absolute points over CoT and 22.72 points over CoT-SC.

Universal Applicability Across Model Types. A key goal of our work was to develop a universal framework effective for all types of models. The results validate this design: ACS provides improvements for both specialized "reasoning" models and general-purpose "non-reasoning" models. For instance, `SmolLM3-3B (Thinking`

		Reasoning Model		Non-Reasoning Model	
	Variant	**Accuracy (%)**	**Δ vs Full**	**Accuracy (%)**	**Δ vs Full**
principles	w/o Signal Purification	14.28	-28.57	33.33	-27.78
Aggregation	*Layer Agg:* Average + Linear Decay	30.77	-12.08	55.56	-5.55
	Layer Agg: Attention Rollout	21.43	-21.42	44.44	-16.67
	Layer Agg: Average (Ours)	**42.85**	–	**61.11**	–
	Head Agg: Max Attention	35.72	-7.13	50.00	-11.11
	Head Agg: Mean Attention	38.46	-4.39	50.00	-11.11
	Head Agg: Kurtosis (Ours)	**42.85**	–	**61.11**	–
Diffusion	w First Order Kernel (no propagation)	21.43	-21.42	33.33	-27.78
	w Second Order Kernel	28.57	-14.28	38.89	-22.22
	w Resolvent Diffusion Kernel	35.71	-7.14	38.89	-22.22
	w Heat Kernel (Ours)	**42.85**	–	**61.11**	–

Table 3: We conducted additional analysis on our design with GPQA-DIAMOND data using `DeepSeek-Qwen-1.5B` (reasoning model) and `LLaMA-3.1-8B` (non-reasoning model).

`Mode`) benefits substantially, reaching 90.72% on STRATEGYQA, up from 61.86% with CoT and 76.29% with CoT-SC. Meanwhile, `ACS` also boosts non-reasoning models, such as `SmolLM3-3B (General Mode)`, which improves on STRATEGYQA from 61.54% (CoT) and 69.23% (CoT-SC) to 76.92%. These results demonstrate that our `MIT`-based diagnostic can identify meaningful critical junctures regardless of whether reasoning is explicit or emergent, making `ACS` a general-purpose enhancement for LLM reasoning.

Scalability and Model Size. The performance improvements scale consistently across models of different sizes. The results suggest that the larger the model, the larger the absolute improvement. `LLaMA-3.1-8B` benefits most, with `ACS` boosting its overall accuracy by +10.2 points compared to CoT and +7.9 points compared to CoT-SC. In contrast, the smallest model, `Gemma-3-270M`, still sees consistent but more modest gains of +5.3 points over CoT and +3.5 points over CoT-SC. These results show the scalability and broad applicability of our approach.

6 Analysis

6.1 Effect of Influence Principles

By following the influence principles in §3.2, we introduce *Stage 1: Signal Purification* in the `MIT` method, which removes attention to non-semantic tokens and self-attention. Disabling this stage results in a dramatic drop in performance: **-28.57** accuracy points for the reasoning model and **-27.78** for the non-reasoning model. This highlights the importance of filtering noisy or non-semantic dependencies before applying structural analysis to the attention graph and the effectiveness of our influence principles. In fact, the introduction of influence principles provides the most performance gain compared to aggregation strategies and diffusion process. More work on the purification effectiveness can be found in Appendix A.7.

6.2 Effect of Aggregation Strategies

We evaluate both layer-wise and head-wise aggregation strategies. For layer aggregation, our simple average pooling significantly outperforms alternatives: applying linear decay leads to a **-12.08** drop in the reasoning model, and using Attention Rollout results in a **-21.42** reduction. This suggests that averaging captures structural signals more robustly than decay or recursive schemes. On the head level, our kurtosis-based aggregation, which quantifies sharpness and consensus, consistently outperforms mean or max attention heuristics. Replacing kurtosis with max pooling leads to **-7.13** degradation (reasoning) and **-11.11** (non-reasoning), while mean pooling performs only slightly better. These results affirm the importance of capturing the distributional shape of attention, rather than relying on scalar summaries alone. More study on aggregation strategies can be found in Appendix A.7.

6.3 Effect of Diffusion Processes

We evaluate the impact of the diffusion module (§3.3) by varying the choice of propagation kernel. Restricting diffusion to a first-order kernel (i.e., no multi-hop propagation) results in a sharp perfor-

mance drop, **-21.42** points on the reasoning model and **-27.78** on the non-reasoning model. This introduces the largest performance drop among all kernels, highlighting the importance of propagation. Extending to a second-order kernel partially recovers performance but remains inferior to the full model. Among deeper kernels, the resolvent (Neumann) kernel outperforms the shallow variants yet still falls short of the heat kernel. The heat kernel (matrix exponential) consistently achieves the best results across both model types. This pattern supports that crucial source–target relations are typically captured by short or middle paths, and that influence should decay with path length. By weighting short and long paths more uniformly, the resolvent kernel violates this bias and underperforms. Further analyses are in Appendix A.11.

7 Discussion

Is ACS expensive? To calculate the end-to-end cost for ACS including MIT, we use floating-point operations (FLOPs) to quantify cost since this fully covers the additional computation introduced by MIT. We use FLOPs rather than wall-clock time or LLM call counts deliberately. Wall-clock time is sensitive to parallelism: since ACS generates multiple suffix samples independently, they can be executed in parallel across GPUs, which would make clock-time comparisons artificially favorable to ACS and obscure the true computational cost. FLOPs measure raw computation regardless of how it is scheduled, making the comparison conservative and hardware-independent. LLM call count is similarly inadequate: it counts only model forward passes and ignores the matrix operations in MIT 's influence propagation (aggregation and heat kernel diffusion), which constitute real computational overhead. FLOPs account for all of this. The full detail of our cost calculation method is in Appendix A.8. As shown in Figure 4, we reduce the average computational cost by approximately **-72.86%** for reasoning models and **-51.14%** for non-reasoning models, demonstrating substantial efficiency gains while improving performance. Additional analysis is in Appendix A.8.5.

How precise is MIT? ACS is proven to improve reasoning performance based on insights from MIT. But do the critical positions surfaced by MIT line up with the ones discovered in the rollout study? We label all positions whose correctness ratio exceeds the SC baseline by at least $\tau = 10\%$ as

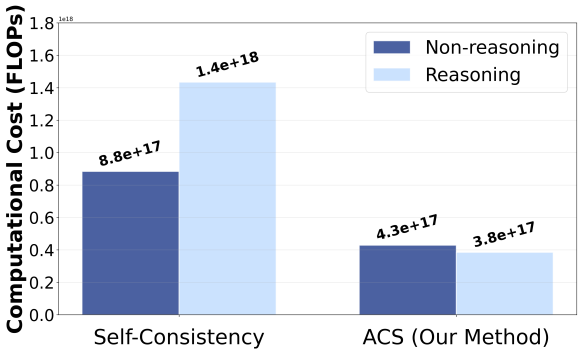


Figure 4: ACS reduces FLOPs by ∼73% for reasoning models and ∼51% for non-reasoning models compared to self-consistency, an average reduction of ∼60%.

critical, and measure how often MIT 's top-1 falls within this set. MIT 's top-1 matches *any* critical position in **86%** of traces, and its top-5 in **91%**. This is the appropriate precision metric: the goal is to identify *a* high-leverage intervention point, not the single global optimum. Attempting to exploit MIT's top-5 coverage by spreading the budget across all five positions consistently *hurts* performance (Figure 13b): diluting samples across positions washes out the majority-voting signal that makes top-1 concentration effective. ACS therefore pairs MIT's high set-level precision with a concentrated single-position budget, delivering substantial accuracy gains while significantly reducing cost.

8 Conclusion

We introduced the concept of **critical positions** in LLM reasoning — locations in reasoning traces where intervention via re-sampling has outsized effect on final outcomes, independent of the specific tokens. We presented empirical evidence that critical positions are real, token-decoupled, and structurally patterned. To identify them at scale, we introduced **Metacognitive Influence Tracing (MIT)**, an unsupervised method that applies a metacognitive lens to reasoning by transforming attention patterns into a propagated influence graph via heat kernel diffusion. Powered by MIT, we introduced **Adaptive Critical Sampling (ACS)**, an intervention framework that exploits critical positions for efficient inference. Across six benchmarks and five models, ACS improves accuracy by an average of +7.78 points over CoT while cutting computation by 59.83%. Together, MIT and ACS show that targeting reasoning at its most intervention-sensitive moments yields more reliable and efficient systems.

9 Limitations

MIT currently relies on explicit, text-based reasoning traces, raising questions about its applicability to tasks where reasoning is implicit. Our experiments focus on multiple-choice and short-answer benchmarks where correctness is well-defined; extending ACS to open-ended generation tasks is a natural next step, and existing automatic evaluation metrics (e.g., BERTScore, LLM-as-judge) provide a tractable path for measuring answer quality in that setting. While ACS is more efficient than brute-force baselines, the MIT analysis itself can be computationally intensive; future work could develop more efficient approximations of influence propagation such as the mitigation method mentioned in Appendix A.9.3. These limitations reflect the scope of the current study rather than fundamental barriers, and we consider them as future opportunities.

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online. Association for Computational Linguistics.

Evan Becker and Stefano Soatto. 2024. Cycles of thought: Measuring LLM confidence through stable explanations. *arXiv preprint arXiv:2406.03441*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pages 17682–17690.

Eric Bigelow, Ari Holtzman, Hidenori Tanaka, and Tomer Ullman. 2025. Forking paths in neural text generation. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*.

Hila Chefer, Shir Gur, and Lior Wolf. 2021. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 782–791.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2024. Do NOT think that much for 2+3 = ? on the overthinking of o1-like LLMs. *arXiv preprint arXiv:2412.21187*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Łukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Allan M. Collins and Elizabeth F. Loftus. 1975. A spreading-activation theory of semantic processing. *Psychological Review*, 82(6):407–428.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578, Bangkok, Thailand. Association for Computational Linguistics.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. 2025. When attention sink emerges in language models: An empirical view. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 1 others. 2025. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8154–8173.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 22199–22213.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

Zicheng Lin, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. 2025. Critical tokens matter: Token-level contrastive estimation enhances LLM's reasoning capability. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 37906–37918. PMLR.

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. 2025. Can 1B LLM surpass 405B LLM? rethinking compute-optimal test-time scaling. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*.

Maxwell-Jia. 2024. AIME_2024: American invitational mathematics examination 2024 (I & II). Hugging Face Datasets.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 14014–14024.

Jianfeng Pan, Senyou Deng, and Shaomang Huang. 2025. CoAT: Chain-of-associated-thoughts framework for enhancing large language models reasoning. *arXiv preprint arXiv:2502.02390*.

Chen Qian, Dongrui Liu, Haochen Wen, Zhen Bai, Yong Liu, and Jing Shao. 2025. Demystifying reasoning dynamics with mutual information: Thinking tokens are information peaks in LLM reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling (COLM)*.

John H. Reynolds and Robert Desimone. 2003. Interacting roles of attention and visual salience in V4. *Neuron*, 37(5):853–863.

Jacqueline S. Sachs. 1967. Recognition memory for syntactic and semantic aspects of connected discourse. *Perception & Psychophysics*, 2(9):437–442.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain-of-thought reasoning in language models. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*.

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

Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2026. When more is less: Understanding chain-of-thought length in LLMs. In *Proceedings of the Fourteenth International Conference on Learning Representations (ICLR)*.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025. TokenSkip: Controllable chain-of-thought compression in LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3351–3363, Suzhou, China. Association for Computational Linguistics.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 11809–11822.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*.

A Appendix

Appendix Contents

A.1 Related Work . . . 14
A.2 Experiment Setup . . . 15
A.2.1 Datasets and task taxonomy. . . . 15
A.2.2 Hyper Parameters and Additional Details . . . 15
A.3 The MIT Methodology Algorithm . . . 16
A.4 Threshold Sensitivity for Critical Position Definition . . . 17
A.5 Critical Position Study . . . 17
A.5.1 Cognitive Marker Distribution . . . 17
A.5.2 Critical Position Distance Study . . . 18
A.6 Propagation Parameter . . . 18
A.7 Layer wise Aggregation Analysis . . . 19
A.8 Cost Calculation . . . 19
A.8.1 Cost of LLM Generation . . . 20
A.8.2 Cost of the MIT Diagnostic . . . 20
A.8.3 Proposed Cost Comparison Framework . . . 21
A.8.4 Theoretical Cost Analysis . . . 22
A.8.5 Empirical Cost Analysis . . . 22
A.9 Memory Consumption Calculation . . . 23
A.9.1 The Memory Consumption of MIT . . . 23
A.9.2 Mitigation 1: Layer-by-Layer Calculation . . . 23
A.9.3 Mitigation 2: Reducing Complexity to $O(K)$ . . . 23
A.9.4 Experiments . . . 23
A.10 Additional Experiments . . . 25
A.11 Propagation Kernels . . . 26
A.11.1 Theoretical Analysis . . . 26
A.11.2 Empirical Analysis . . . 26
A.12 Prompts . . . 29
A.13 Examples . . . 30
A.13.1 Example for MIT . . . 30
A.13.2 Example for ACS with Reasoning Model . . . 31
A.13.3 Example for ACS with Non-Reasoning Model . . . 32
A.13.4 Failure Example for ACS . . . 33
A.14 Potential Impact and Future Works . . . 34
A.15 Use of LLMs . . . 34

A.1 Related Work

Enhancing Chain-of-Thought (CoT) Since *Chain-of-Thought* (CoT) prompting was popularized for step-wise reasoning, a large body of work has explored how to elicit, structure, and aggregate intermediate thoughts. This work usually applies to both non-reasoning models and reasoning models. Zero-shot and few-shot CoT show that lightweight hints can bootstrap decomposition without heavy finetuning (Kojima et al., 2022; Wei et al., 2022). *Self-Consistency* (SC) improves reliability by sampling diverse reasoning paths and voting on answers (Wang et al., 2023). Beyond linear traces, researchers proposed richer structures: *Tree-of-Thoughts* (ToT) performs breadth/depth exploration with value-guided selection (Yao et al., 2023), and *Graph-of-Thoughts* connects partial solutions and subproblems in a general graph to enable reuse (Besta et al., 2024). Task-decomposition templates such as *Least-to-Most* and complexity-aware prompting steer models toward simpler subgoals (Zhou et al., 2023; Fu et al., 2023).

Enhancing LLM Reasoning Capability Beyond formatting thoughts, several threads aim to improve inference time reasoning itself, often referred to as Test Time Scaling. This line of work usually applies to models with reasoning capabilities. Test-time compute scaling strategies increase sample count, depth, or verification steps, often coupled with selective aggregation and verification to filter inconsistent traces (Cobbe et al., 2021; Dhuliawala et al., 2024). CoAT encourages multiple complementary perspectives before convergence (Pan et al., 2025). Other studies probe how far we can push "small" models with better training, curricula, and targeted heuristics, sometimes surpassing naively prompted much larger models (Liu et al., 2025). At the same time, multiple papers warn that "more thinking" can backfire: both controlled studies and empirical audits document overthinking and propose inference-time policies to cap or adapt CoT length (Chen et al., 2024; Cuadron et al., 2025). Wu et al. (2026) uncover an inverted-U relation between CoT length and accuracy, with optimal lengths growing with task difficulty but shrinking with model capability; RL training further biases models toward shorter, more efficient chains (Wu et al., 2026). Complementary work finds that excessively long CoTs can harm performance on simple instances, cautioning against unbounded test-time scaling (Chen et al., 2024; Cuadron et al., 2025; Xia et al., 2025).

Attention Attribution Recent work has examined whether attention weights can serve as faithful explanations, but evidence shows that raw attention distributions are often unreliable and can be manipulated without changing predictions (Serrano and Smith, 2019; Jain and Wallace, 2019). Studies of BERT and related models reveal recurring head patterns linked to syntax and coreference, while many heads appear redundant or dispensable (Clark et al., 2019; Voita et al., 2019; Michel et al., 2019). To better capture information flow, techniques such as attention rollout and relevance propagation have been proposed to trace or aggregate attention through layers, yielding more faithful token-level attributions (Abnar and Zuidema, 2020; Chefer et al., 2021). In parallel, probing methods such as classifier probes, edge probing or structural probes, demonstrate that specific layers encode linguistic features in a pipeline-like progression (Hewitt and Manning, 2019; Tenney et al., 2019). More recently, researchers have discovered "sink tokens" that absorb disproportionate attention mass, leading to diluted signals across depth and biased rollout analyses, with mitigation strategies explored in both language and vision Transformers (Kobayashi et al., 2020; Gu et al., 2025).

Critical Tokens Most closely related, Lin et al. (2025) introduce the concept of critical tokens — elements within reasoning trajectories whose lexical content significantly influences outcomes. Their method identifies these tokens via cross-trajectory contrastive estimation, requiring two fine-tuned models and ground-truth correctness labels. Our work introduces a complementary concept, critical positions, which differs in three substantive respects. First, critical positions are defined by structural rather than lexical properties: our analysis in §2 finds that token changes at identified positions correlate weakly with answer changes, indicating these positions are characterized by their role in the trace rather than by specific tokens. Second, `MIT` operates on a single unlabeled trace using only attention structure, with no supervision or contrastive trajectories. Third, `ACS` is an inference-time intervention, complementary to training-time alignment approaches such as Lin et al.'s cDPO. Critical tokens and critical positions are related but distinct phenomena, opening parallel lines of inquiry into where and how reasoning trajectories can be intervened upon.

A.2 Experiment Setup

A.2.1 Datasets and task taxonomy.

To comprehensively evaluate reasoning across diverse domains, we group benchmarks by the primary reasoning skill they target. This taxonomy spans logical, arithmetic, commonsense, and symbolic reasoning, ensuring that our analysis captures a wide spectrum of cognitive capabilities.

- **Logical reasoning: GPQA-DIAMOND** (Rein et al., 2024). GPQA-DIAMOND is a recently introduced benchmark consisting of graduate-level, Google-proof questions designed to probe advanced logical reasoning. The dataset emphasizes multi-hop reasoning and deductive inference, requiring models to integrate evidence across multiple statements before arriving at a conclusion. Unlike typical factual QA benchmarks, GPQA-DIAMOND minimizes reliance on web-searchable facts, thereby isolating genuine reasoning ability from memorization.

- **Arithmetic reasoning: MATH500 and AIME24** (Hendrycks et al., 2021; Lightman et al., 2024; Maxwell-Jia, 2024). The arithmetic category evaluates the capacity of models to manipulate numbers, apply mathematical formulas, and perform multi-step computations. MATH500 is a curated subset of 500 problems sampled from the broader MATH dataset (Hendrycks et al., 2021), covering algebra, geometry, probability, and other competition-style mathematics. AIME24 is a collection of problems from the 2024 American Invitational Mathematics Examination, one of the most challenging pre-collegiate contests. The questions are designed to require multi-step derivations and precise symbolic manipulation, often beyond simple arithmetic.

- **Commonsense reasoning: STRATEGYQA** (Geva et al., 2021). STRATEGYQA is a benchmark focusing on implicit commonsense reasoning. Unlike direct QA datasets, the questions often cannot be answered with a single fact but instead require models to generate a strategy for combining multiple weak signals (e.g., "Could a crocodile run a marathon?"). This dataset tests a model's ability to operationalize background knowledge and apply everyday reasoning skills in novel contexts.

- **Symbolic reasoning: COINFLIP and LAST LETTER** (Wei et al., 2022). These synthetic tasks were introduced to stress-test chain-of-thought prompting. In COINFLIP, the model must track the state of a coin after a sequence of flips, which requires deterministic state updates over potentially long sequences. In LAST LETTER, the model must return the last letter of each word in a given string and concatenate them. Despite their apparent simplicity, these tasks are challenging for LLMs, as they require precise symbolic manipulation, memory of intermediate steps, and robustness to input length.

Together, these datasets form a comprehensive suite that spans from formal logic and competition-level mathematics to real-world commonsense and strict symbolic reasoning. This breadth allows us to evaluate whether improvements from our proposed method generalize across fundamentally different reasoning paradigms. For ablation study, we select the first 50 questions for GPQA-Diamond as testing dataset.

A.2.2 Hyper Parameters and Additional Details

For baselines we compare with CoT and Self-Consistency with 9 samples, denoted CoT-SC. In the sampling process for ACS, we keep the prefix fixed and generate additional 8 candidates. This ensures that our approaches and Self-Consistency baseline have the same total number of samples.

For the max token length, we follow the practice of (Liu et al., 2025) to set the max token to be 8192 for all the models except for LLaMA-3.1-8B, where we set the max token to 4096. For the sampling process, we set the max token uniformly to 4096.

For smaller models, these settings cause some of the questions not to be answered as the thinking process does not finish. For such cases, we exclude them from the calculation of the final accuracy as we consider this a model limitation, not a limitation of our methods.

We set the temperature to be 0.7 throughout all the models in all the generation processes for both our method and baselines.

We consider punctuations, special tokens and stop words from nltk as non-semantic tokens.

A.3 The MIT Methodology Algorithm

For the full procedures for the MIT method, please refer to the algorithm below.

Algorithm 2 Metacognitive Influence Tracing (MIT)

Require: Raw attention matrices $\{A_{l,h}\}_{l=1..L,\, h=1..H}$ with $A_{l,h} \in \mathbb{R}^{K\times K}$, token sequence T, answer token indices T_{ans}, CoT token indices T_{CoT}, diffusion scale $\gamma > 0$

procedure MIT($\{A_{l,h}\}, T, T_{\text{ans}}, T_{\text{CoT}}$)

// Stage 1: Attention Signal Purification

  Initialize purified set $\{A'_{l,h}\} \leftarrow \varnothing$

  for $l = 1$ to L **do**

    for $h = 1$ to H **do**

      $A_{\text{temp}} \leftarrow A_{l,h}$

      $A_{\text{temp}} \leftarrow \textsc{FilterSemanticTokens}(A_{\text{temp}}, T)$

      $\text{diag}(A_{\text{temp}}) \leftarrow \mathbf{0}$ ▷ remove self-attention

      Add A_{temp} to $\{A'_{l,h}\}$

    end for

  end for

// Stage 2: Layer Aggregation

  Initialize $A'' \in \mathbb{R}^{H\times K\times K}$ with zeros

  for $h = 1$ to H **do**

    $A''_h \leftarrow \frac{1}{L}\sum_{l=1}^{L} A'_{l,h}$

  end for

// Stage 3: Kurtosis-Weighted Head Aggregation

  Initialize $A \in \mathbb{R}^{K\times K}$ with zeros

  for $i = 1$ to K **do** ▷ target token index

    $w \in \mathbb{R}^{H}$ ▷ per-head weights for row i

    for $h = 1$ to H **do**

      $w_h \leftarrow \text{Kurtosis}(A''_h[i,:])$

    end for

    $w \leftarrow w / \sum_{h=1}^{H} w_h$ ▷ normalize

    for $j = 1$ to K **do** ▷ source token index

      $A[i,j] \leftarrow \sum_{h=1}^{H} w_h \cdot A''_h[i,j]$

    end for

  end for

// Stage 4: Influence Propagation

  $S \leftarrow \exp(\gamma A)$ ▷ matrix exponential diffusion over token graph

// Stage 5: Critical Position Identification

  Initialize $\text{Score} \in \mathbb{R}^{|T_{\text{CoT}}|}$ with zeros

  for each token index $j \in T_{\text{CoT}}$ **do**

    $\text{Score}(j) \leftarrow \sum_{i\in T_{\text{ans}}} S[i,j]$

  end for

  $j_{\text{critical}} \leftarrow \arg\max_{j\in T_{\text{CoT}}} \text{Score}(j)$

  return j_{critical}

end procedure

A.4 Threshold Sensitivity for Critical Position Definition

The definition of a critical position requires a threshold τ: a position qualifies as critical if its correctness ratio — the fraction of rollout continuations that yield a correct answer — exceeds the SC correctness ratio at equal budget by at least τ (relative). We study how the choice of τ affects two quantities across 50 incorrect GPQA-DIAMOND traces per model: (1) the fraction of traces with at least one critical position (**coverage**), and (2) the average number of critical positions per qualifying trace.

	DeepSeek-Qwen-1.5B		LLaMA-3.1-8B	
τ	Coverage	Avg. # crit. pos.	Coverage	Avg. # crit. pos.
10%	100%	5.50	100%	3.30
30%	45%	1.80	58%	1.29
50%	12%	1.25	32%	1.09

Table 4: Threshold sensitivity on GPQA-DIAMOND. Coverage = fraction of traces with $\geq$1 critical position. Avg. # crit. pos. = average number of critical positions per qualifying trace.

Why $\tau = 10\%$? The data reveal a fundamental trade-off. Higher τ yields fewer, higher-quality critical positions per trace, but at the cost of coverage. At $\tau = 30\%$, coverage drops to 45% and 58% — meaning over half of traces have no critical position at all. At $\tau = 50\%$, coverage collapses to 12% and 32%, and the average count per qualifying trace converges to near-singular (1.25 and 1.09), making critical positions a rare exception rather than a structural property. We choose $\tau = 10\%$ because **coverage is the primary requirement**: a definition that leaves the majority of traces without a critical position cannot support a general-purpose intervention framework like ACS. At $\tau = 10\%$, every incorrect trace has at least one critical position (coverage = 100%), grounding critical positions as a universal property of reasoning trajectories. The higher average count per trace (5.50 for DeepSeek, 3.30 for LLaMA) reflects that many positions offer meaningful improvement over SC — MIT 's role is precisely to discriminate among them and surface the most influential one.

Does $\tau = 10\%$ hold up where ACS shows weak gains? Table 2 contains rows where ACS delivers little or no improvement over CoT-SC. None of these implicate the threshold choice, for a simple reason: τ is a labeling parameter for the rollout-based oracle; it does not enter ACS's inference-time pipeline. Raising τ would shrink the validation subset but leave MIT 's per-trace selection — and therefore every entry in Table 2 — entirely unchanged. The weak-gain rows are explained by three orthogonal factors. Saturated benchmarks (SmolLM3-Think at 100.00 on three tasks) are pure ceiling effects where no method can improve. AIME24 rows ($n \approx 30$) lie within the noise envelope of small-sample evaluation, where a single question shifts accuracy by $\sim$3.3 points. The Gemma-270M outlier reflects model-scale signal degradation: MIT relies on attention being a clean structural signal, and at 270M parameters that signal weakens regardless of how critical positions are defined.

A.5 Critical Position Study

A.5.1 Cognitive Marker Distribution

We study the vocabulary of reasoning and non-reasoning models that are surfaced by MIT , and present their distributions grouped by model in Figure 5. Across both reasoning and non-reasoning models, we observe consistent reliance on a small set of cognitive markers. Words such as *"okay"* and *"step"* appear as dominant tokens across multiple models, highlighting their role as universal markers of process control and explicit reasoning progression.

Interestingly, the *SmoLM3-3B (Thinking)* model exhibits a strikingly uniform pattern: all critical positions are marked by the token *"okay"*, which is also found in the reasoning-oriented *DeepSeek-Distilled-Qwen-1.5B*. This suggests that these models adopt a highly cognitive style of reasoning, repeatedly signaling checkpoints of internal validation. However the particular word used differs by model and can be due to the training process of each model, such as SmoLM3-3B (Thinking) preferring *"okay"* whereas DeepSeek-Distilled-Qwen-1.5B prefers *"wait"*.

By contrast, non-reasoning models such as *Gemma-3-270M-IT* and *LLaMA-3.1-8B* reveal more heterogeneous distributions (Figure 5). Tokens such as *"answer"*, *"therefore"* surface as frequent critical positions, reflecting a more linguistically diverse and procedural reasoning style. The particular word *"step"* is given in the prompt, so it might not be consistent when the prompt changes. But other words that connect reasoning together such as *"therefore"*, *"because"*, *"given"* do not appear within the prompts.

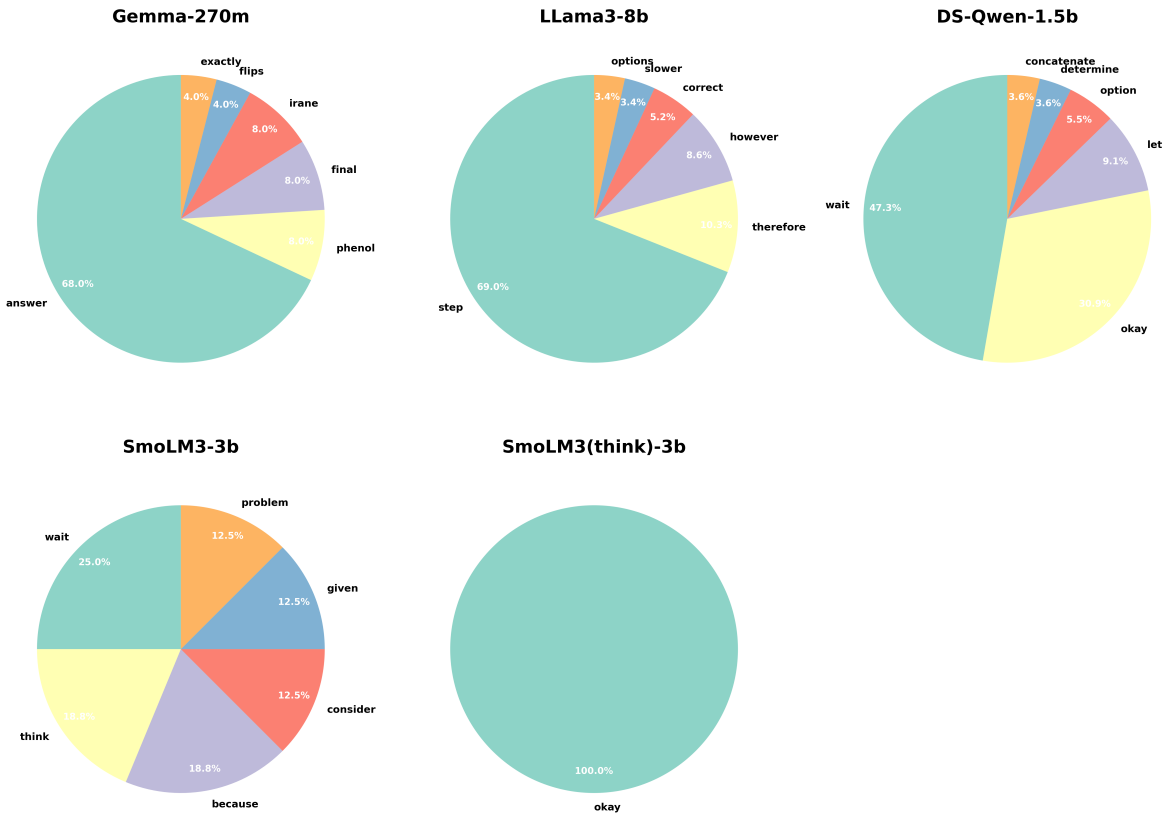


Figure 5: The distribution of cognitive markers surfaced as critical positions by models.

These findings suggest that reasoning models converge toward concentrated cognitive markers to signal decision checkpoints, while non-reasoning models spread cognitive load across diverse linguistic markers. This contrast highlights a fundamental difference in how model classes manage and externalize their reasoning processes.

A.5.2 Critical Position Distance Study

We study the distance between the critical position surfaced by `MIT` and the start of the answer token to gauge the diffusion scope `MIT` provides. As shown in Figure 6, our analysis reveals distinct cognitive patterns across different language models in terms of how far their most influential positions are located from the final answer. Generally reasoning models exhibit longer range. We hypothesize this capability allows the model to understand previous trace content and leads to better performance.

For non-reasoning models, Gemma-3-270M-IT demonstrates the most localized reasoning strategy, suggesting it relies heavily on immediate contextual cues near the answer location. LLaMA-3.1-8B and SmolLM3-3B exhibit similar distributed reasoning patterns with median distances of 74.5 and 77 tokens respectively, though their high standard deviations (1,534 and 2,462 tokens) indicate significant variability in their reasoning strategies.

For reasoning models, DeepSeek-Distilled-Qwen-1.5B has median distance of 492 tokens, showing more structured long-range reasoning than the non-reasoning models but remaining more focused than the bigger reasoning models. SmolLM3-3B(Think), the specialized reasoning variant, shows the most systematic pattern with a median distance of 780 tokens — demonstrating that explicit reasoning training encourages models to establish critical decision points much earlier in the thought process. The relatively low variance in SmolLM3-3B(Think) (502 tokens) compared to its base model suggests that reasoning-trained models develop more consistent cognitive strategies, while non-reasoning models exhibit more unpredictable patterns depending on the specific question type.

A.6 Propagation Parameter

Selecting the Propagation Parameter γ To better understand the influence of the exponential decay parameter γ, we visualized the rank distribution across a range of values: $\gamma \in \{1, 2, 3, 5, 8, 13\}$. As shown in Figure 7, smaller values of γ (e.g., 1 and

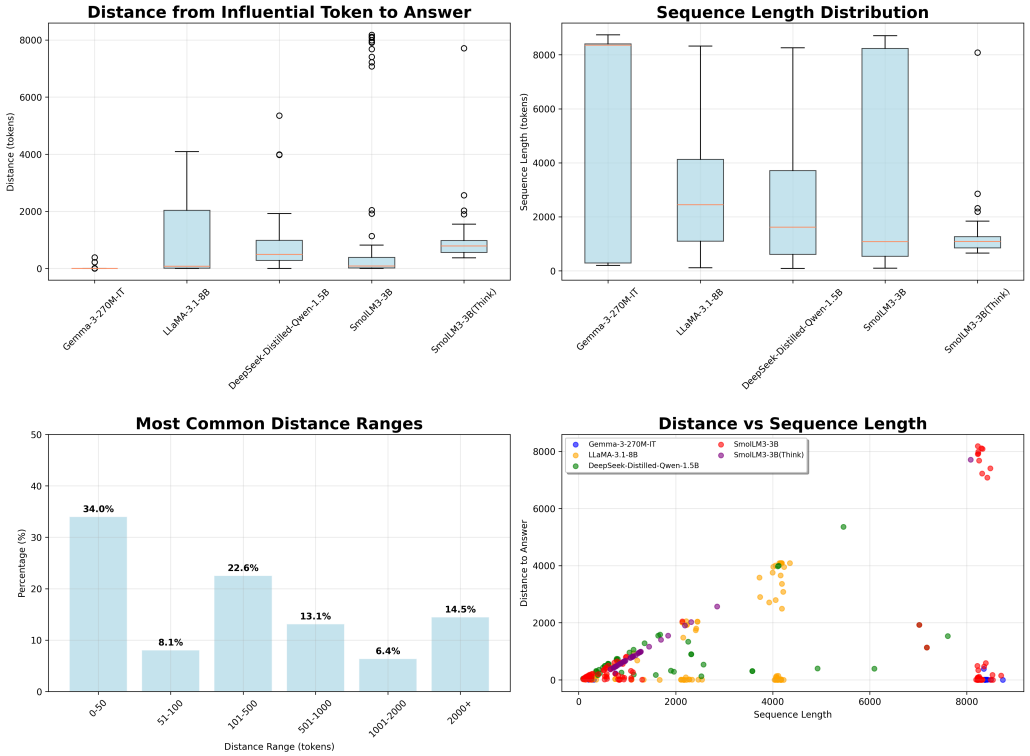


Figure 6: The distance between the surfaced critical position by MIT and the start of the answer token. Our analysis reveals distinct cognitive patterns across different language models in terms of how far their most influential positions are from the final answer. Generally reasoning models exhibit longer range. We hypothesize this capability allows it to understand previous trace content and lead to better performance.

2) emphasize higher influence concentration in the mid- to high-ranked tokens (i.e., rank 100–1000) in the distance between 50 to 1000, while preserving noticeable gradients across distance. In contrast, higher values of γ (e.g., 5, 8 or 13) over-emphasize long-distance influence, resulting in overly flat and uniform rank maps between the distance 50 to 1000, indicative of diminished propagation power. Based on these insights, we selected $\gamma = 2$ as the default for exponential propagation, as it provides a good trade-off between sparsity and depth-aware diffusion.

This choice is further supported by the ablation results in Figure 8. For non-reasoning models, performance consistently declines as γ increases beyond 2, while for reasoning models we observe an unexpected spike at $\gamma = 13$. These differences suggest that reasoning and non-reasoning models occupy distinct optimization regimes, and that the optimal γ may depend on the model's reasoning style rather than a single universal setting. During our experiments for the main result, we did not use different settings for different models nor datasets. The results thus present the universal performance under the same un-tuned setting for a broad range of tests and models.

A.7 Layer wise Aggregation Analysis

We conducted a layer-wise aggregation analysis to study the impact of different methods on information propagation across layers, for both reasoning and non-reasoning models as seen in Figure 9. Our findings are consistent across model types, and we summarize the key insights as follows:

- **Cleaning** improves interpretability by isolating semantically meaningful dependencies, allowing logical influence patterns to emerge more clearly.
- **Average** and **Average Decay** aggregation strategies highlight fine-grained distinctions across layers, enabling the emergence of more nuanced, structured patterns.

A.8 Cost Calculation

To provide a fair and rigorous comparison between Self-Consistency (SC) and our proposed Adaptive Critical Sampling (ACS), we must account for the full computational cost of each method. Wall-clock time rewards parallelism and would make MIT's parallelizable diagnostic pass appear cheaper than it is. Other methods like API calls do not account for the computation MIT has to conduct.

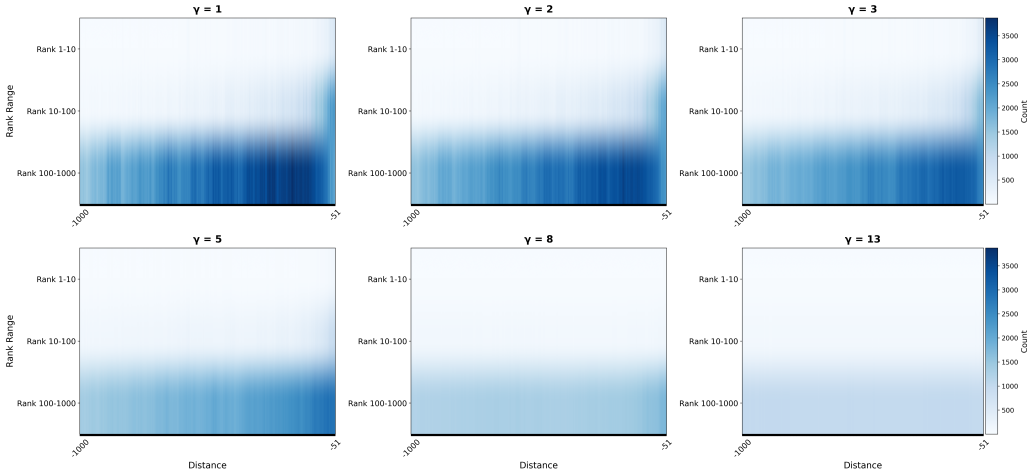


Figure 7: Rank distribution across different exponential decay values ($\gamma \in \{1, 2, 3, 5, 8, 13\}$).

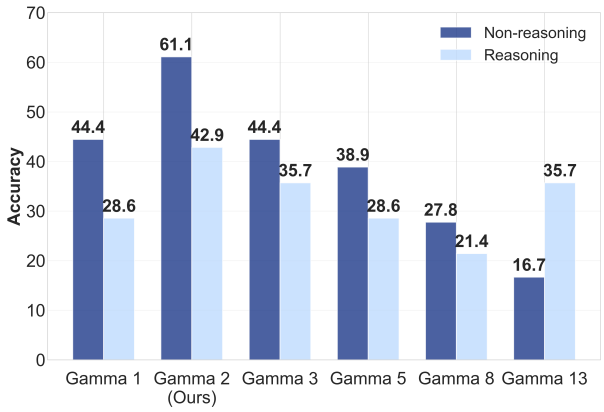


Figure 8: ACS performance under different gamma values: ($\gamma \in \{1, 2, 3, 5, 8, 13\}$)

Thus we propose using **Floating-Point Operations (FLOPs)** as a unified currency to measure both LLM inference and the numerical computations within MIT. The total cost for any method can be expressed as:

$$\text{TotalCost}_{\text{method}} = \text{Cost}_{\text{diagnostic}} + \text{Cost}_{\text{generation}} \tag{7}$$

For Self-Consistency, $\text{Cost}_{\text{diagnostic}}$ is zero. For ACS, it is the cost of running the MIT algorithm.

A.8.1 Cost of LLM Generation

The computational cost of autoregressive generation from a transformer model is dominated by the forward passes. A common approximation for the FLOPs required for a single forward pass is $2 \times P \times T$, where P is the number of model parameters and T is the sequence length. Since generation is sequential, the total cost for a sequence of length T_{total} is the sum of the costs for generating each token:

$$\text{Cost}_{\text{gen}}(T_{\text{total}}) \approx \sum_{i=1}^{T_{\text{total}}} 2 \times P \times i \approx P \times T_{\text{total}}^2 \tag{8}$$

A.8.2 Cost of the MIT Diagnostic

The cost of the MIT diagnostic is non-trivial and can be broken down by its most computationally intensive stages, operating on a sequence of length K:

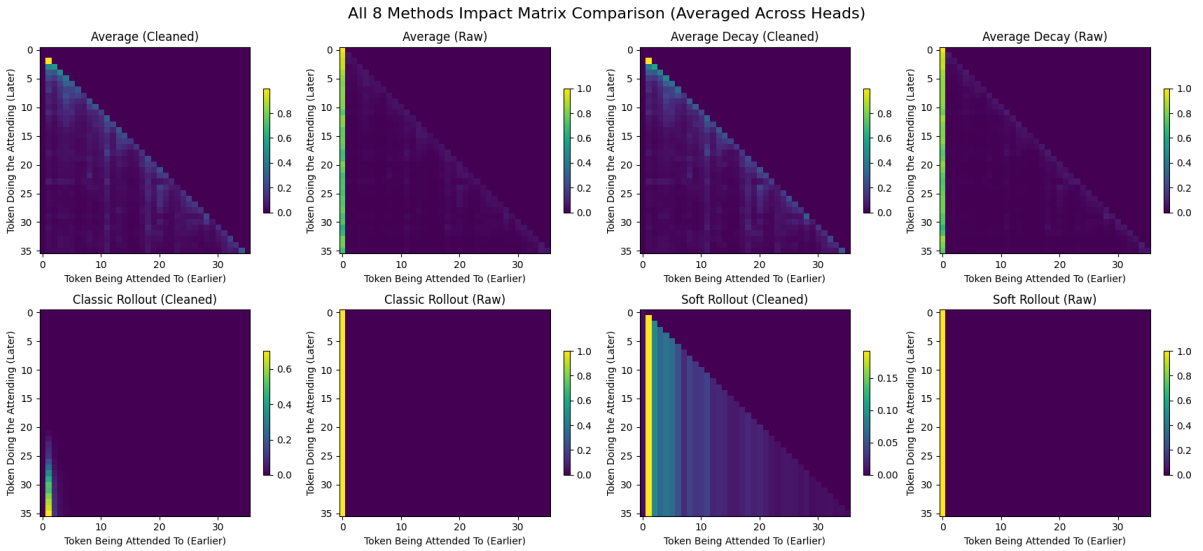
- **Aggregation (Stages 1-3):** These stages involve matrix manipulations across all layers (L) and heads (H) for the $K \times K$ attention matrices. The cost is roughly proportional to the square of the sequence length:

$$\text{Cost}_{\text{agg}} \propto L \times H \times K^2 \tag{9}$$

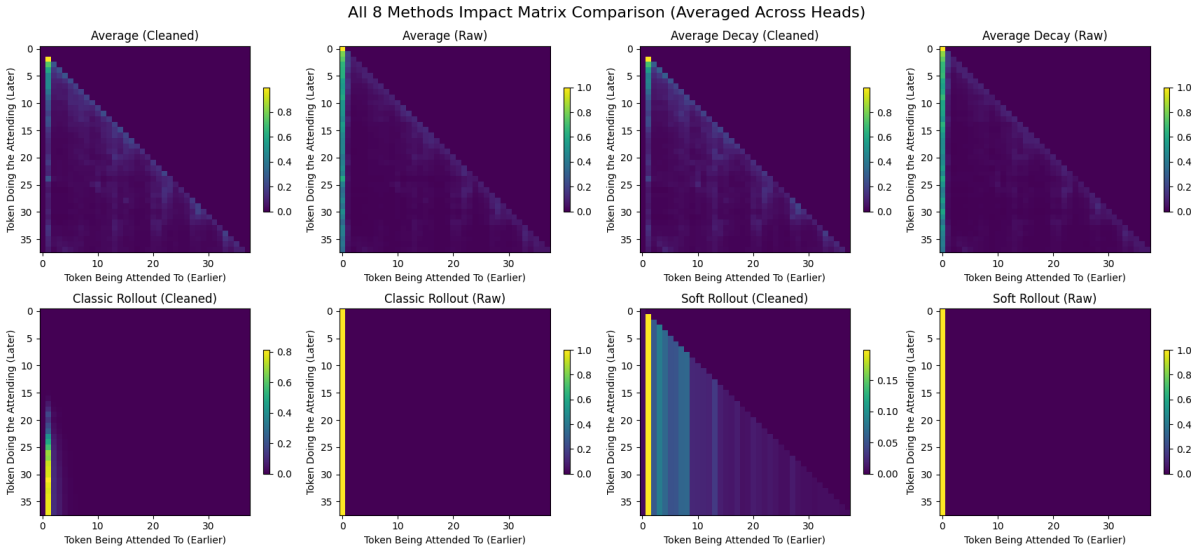
- **Propagation (Stage 4):** This is the primary computational bottleneck. It requires calculating the matrix exponential of the $K \times K$ influence matrix. The complexity of this numerical operation is approximately cubic in the matrix dimension:

$$\text{Cost}_{\text{prop}} \propto K^3 \tag{10}$$

The total cost of MIT is therefore dominated by the propagation step, scaling cubically with the length



(a) LLaMA-3.1-8B



(b) DeepSeek-Distilled-Qwen-1.5B

Figure 9: Layer-wise Aggregation Analysis

of the reasoning trace.

$$\text{Cost}_{\text{MIT}}(K) \approx c_1(L \times H \times K^2) + c_2(K^3) \quad (11)$$

where c_1 and c_2 are machine-dependent constants and we set c_1 to be 8 and c_2 to be 40 based on benchmarking on aws ml.g5.48xlarge instance.

A.8.3 Proposed Cost Comparison Framework

Using this framework, we can define the total cost for generating N samples for each method. Let T_{full} be the length of a full reasoning trace and T_{suffix} be the length of the trace generated after a critical position.

The total cost for **Self-Consistency** is:

$$\text{TotalCost}_{\text{SC}} = N \times \text{Cost}_{\text{gen}}(T_{\text{full}}) \quad (12)$$

The total cost for **Adaptive Critical Sampling** is:

$$\begin{aligned}\text{TotalCost}_{\text{ACS}} = &\underbrace{\text{Cost}_{\text{gen}}(T_{\text{full}})}_{\text{Initial Trace}} \\ &+ \underbrace{\text{Cost}_{\text{MIT}}(T_{\text{full}})}_{\text{Diagnostic Overhead}} \\ &+ \underbrace{(N-1) \times \text{Cost}_{\text{gen}}(T_{\text{suffix}})}_{\text{Resampling Suffixes}} \quad (13)\end{aligned}$$

This framework provides a principled and transparent methodology for evaluating the true efficiency trade-offs, revealing under which conditions (e.g., long vs. short reasoning traces) the diagnostic overhead of `MIT` is justified by the savings in generation cost.

A.8.4 Theoretical Cost Analysis

In practice, the apparent cubic overhead of MIT should be understood in relation to the dominant cost of autoregressive generation. While $\text{Cost}_{\text{MIT}}(K)$ scales as K^3, the generation cost scales as $P \times T^2$, where P is the number of model parameters. For modern LLMs with P in the billions and context lengths in the thousands, $P \gg K$, meaning the per-token matrix multiplications in generation dominate the computational bill. As a result, the one-time K^3 diagnostic is negligible compared to repeatedly multiplying against billions of parameters during token generation. Put differently, the crossover point where $\text{Cost}_{\text{MIT}}(K)$ becomes comparable to Cost_{gen} only arises for very small models (tens to hundreds of millions of parameters) or extremely long traces. For the parameter scales where self-consistency is typically applied (multi-billion parameter LLMs), MIT is therefore asymptotically and empirically cheaper than regenerating full traces, which explains the consistent efficiency gains of ACS over SC.

For example, consider $K = 2048$ tokens and a model with $P = 7 \times 10^9$ parameters. The diagnostic propagation step requires on the order of $K^3 \approx 8.6 \times 10^9$ operations. By contrast, generating a full trace costs approximately $P \times K^2 \approx (7 \times 10^9) \times (4.2 \times 10^6) \approx 3 \times 10^{16}$ operations. Thus, the MIT overhead is roughly six orders of magnitude smaller than the generation cost. This illustrates that, for billion-parameter models and typical reasoning trace lengths, the cubic diagnostic is essentially negligible relative to autoregressive decoding.

A.8.5 Empirical Cost Analysis

Interestingly, the reasoning model demonstrates a lower overall computational cost compared to the average non-reasoning model. This counterintuitive result arises from how critical positions are identified and the computational trade-offs inherent in our Adaptive Critical Sampling (ACS) approach. For example, reasoning models such as *DeepSeek-Distilled-Qwen-1.5B* tend to identify critical positions later in their reasoning traces (at approximately 3,304 tokens). In this case, the diagnostic overhead introduced by MIT (1.51×10^{12} FLOPs) is effectively amortized across substantial resampling savings. As a result, the majority of *DeepSeek-Distilled-Qwen-1.5B*'s total cost (3.52×10^{17} FLOPs) comes from efficient suffix resampling rather than redundant full-sequence generation.

By contrast, non-reasoning models display greater variability in computational efficiency. *LLaMA-3.1-8B* achieves notable savings (61.0% reduction) by identifying critical positions at an optimal 2,225 tokens, whereas *Gemma-3-270M-IT* detects critical positions much earlier (563 tokens). This early detection leads to heavier suffix regeneration costs. Specifically, for *Gemma-3-270M-IT*, our method reduces computational cost from 5.69×10^{17} **FLOPs** under Self-Consistency to 3.89×10^{17} **FLOPs**—a reduction of **31.6%**. However, the early critical position means the MIT diagnostic overhead is less effectively amortized, highlighting how our method's computational efficiency depends critically on the timing of critical position identification.

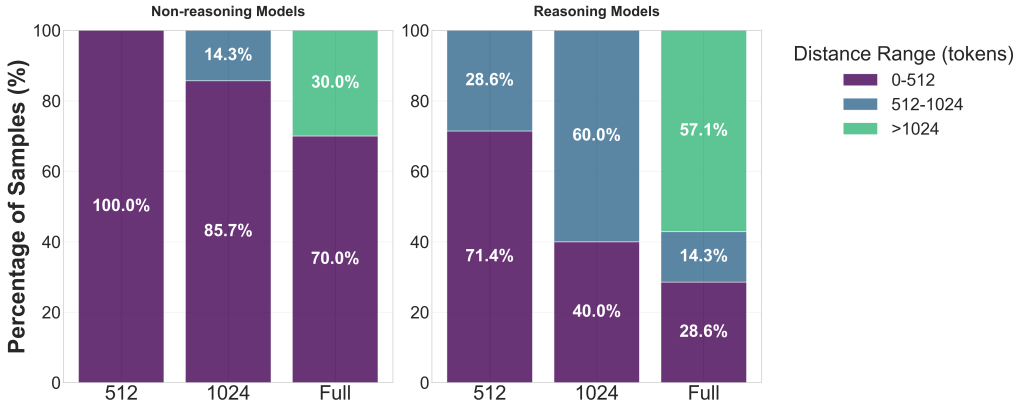


Figure 10: Distance between most influential position and answer token in different window size settings

A.9 Memory Consumption Calculation

A.9.1 The Memory Consumption of MIT

The primary limitation on the scalability of the Metacognitive Influence Tracing (MIT) algorithm is its memory consumption. The core of the method relies on storing and processing full $K \times K$ attention and influence matrices, where K is the length of the reasoning trace. This results in a memory complexity that scales **quadratically**, or $O(K^2)$, with the sequence length, which can become prohibitively large for very long sequences.

A.9.2 Mitigation 1: Layer-by-Layer Calculation

In our application, we mitigate this issue by avoiding the need to load the entire raw attention tensor (of size $L \times H \times K \times K$) into memory at once. Instead, the implementation performs a **layer-by-layer calculation**. It processes the attention matrices for each layer sequentially, updating a running aggregate for each attention head before discarding the raw layer data. This approach reduces the peak memory requirement from $O(L \times H \times K^2)$ down to $O(H \times K^2)$. While this is a significant practical improvement, it does not change the fundamental quadratic scaling with respect to the sequence length K.

A.9.3 Mitigation 2: Reducing Complexity to $O(K)$

To achieve a true linear memory complexity of $O(K)$, a promising future direction is the integration of **sparse attention techniques**, such as a **sliding window** model. In this approach, instead of each token attending to all previous tokens, it would only attend to a fixed number of its immediate neighbors (a "window" of size w). This would reduce the memory required to store attention scores from $O(K^2)$ to $O(K \times w)$, which is linear if the window size w is a constant.

Beyond memory savings, adopting a sliding window also has significant implications for computational efficiency. In the current MIT framework, the diffusion step requires computing the matrix exponential of a dense $K \times K$ influence matrix, leading to cubic cost $O(K^3)$ as listed in Equation equation 11. By restricting attention to a banded sparse structure with bandwidth w, the propagation can be reformulated using sparse linear algebra routines. This reduces the effective cost to approximately $O(K \cdot w^2)$, since each token interacts only within its local neighborhood. When $w \ll K$, this can yield orders-of-magnitude reductions in floating-point operations, especially for long reasoning traces. Thus, sliding window not only lowers memory complexity but also offers a pathway to substantially reduce the computational overhead of MIT.

Informing Sliding Window Size with Research Findings The critical challenge with a sliding window approach is selecting an appropriate window size, as one that is too small could fail to capture important long-range dependencies in the reasoning process. Our findings in Figure 6, which is the analysis of the distance between critical positions and the final answer, reveals that while a significant portion (over 20%) depend on much longer-range connections, sometimes exceeding 2000 tokens, the majority of influential interactions (over 60%) occur within a 500-token range.

A.9.4 Experiments

Based on findings from Figure 6, we have conducted two experiments with the window size set as 512, 1024 respectively. The results are shown in Figure 11. In both settings we do observe a drop

in accuracy, as the reduced window size limits the range of influential position searching.

However we do notice that our propagation method with heat kernel enables positions beyond the window size to be surfaced. This is supported by Figure 10 where we observe the most influential position beyond the window size. For 28.6% of cases for window size 512 for reasoning models, the distance between most influential position and the answer is over 512. We hypothesize that tuning the propagation scale such as the gamma value can help with the propagation of influential positions and leave it as future work.

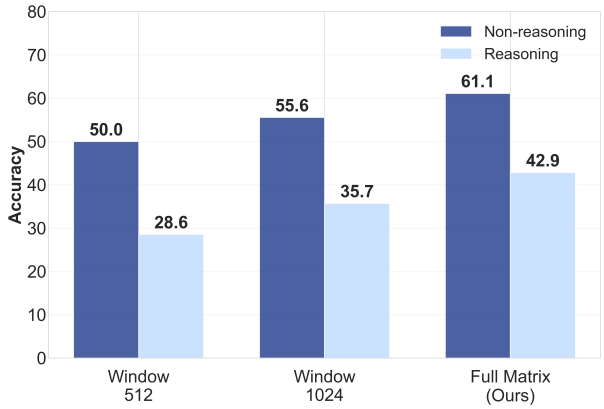


Figure 11: Sliding Window Comparison

A.10 Additional Experiments

Does MIT-identified position outperform random resampling with equal budget? A natural question is whether the gains from ACS stem from MIT 's identification of meaningful positions, or simply from the act of resampling anywhere. To answer this directly, we compare three equal-budget conditions (8 samples each): resampling at the MIT -identified position (**MIT Token**), resampling at a uniformly random position (**Random Token**), and resampling at a randomly selected cognitive marker position (**Random Cognitive**). As shown in Figure 12, MIT -identified positions substantially outperform both random baselines: 61.1 vs. 38.9 (non-reasoning) and 42.9 vs. 28.6 (reasoning) for MIT vs. random token selection. Crucially, Random Cognitive performs no better than Random Token (33.3 vs. 38.9 non-reasoning; 28.6 vs. 28.6 reasoning), confirming that the gains are attributable to MIT 's precise localization of structurally significant positions, not to the presence of a cognitive marker per se.

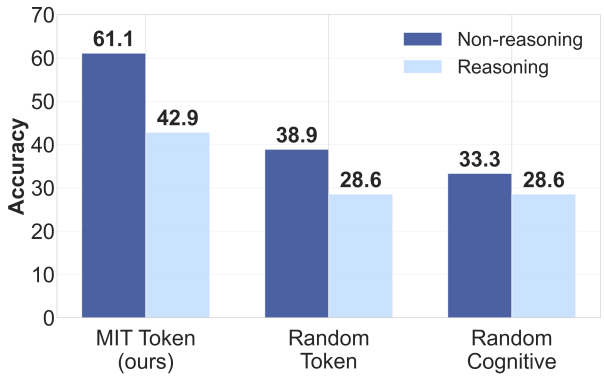
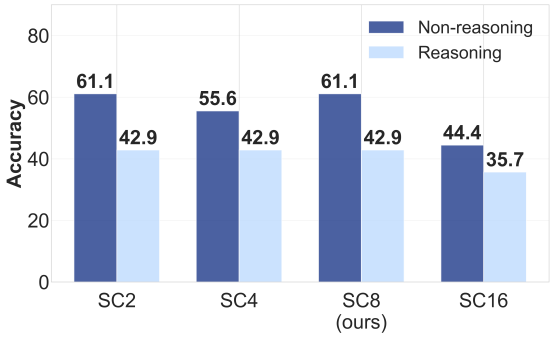
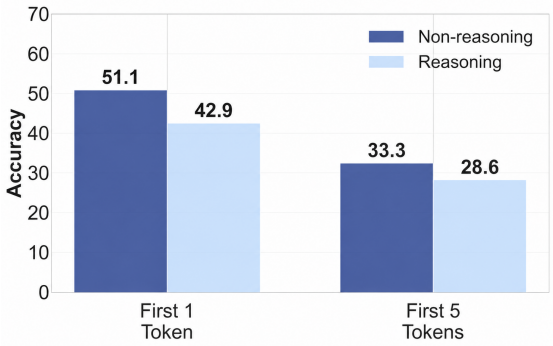


Figure 12: ACS performance with equal sampling budget (8 samples) across three position-selection strategies on GPQA-DIAMOND. MIT-identified positions substantially outperform random position and random cognitive marker baselines for both reasoning and non-reasoning models.

Can we sample less? We conducted additional experiments to study how the number of samples affects the overall accuracy of ACS. We tried sampling 2, 4, 8, 16 times and observe that reducing the number of samples does not necessarily decrease performance, neither does increasing samples improve performance as shown in Figure 13a. Based on the experiment, sampling 2 times will achieve similar effects. However, due to the randomness



(a) Sampling Count Comparison



(b) Sampling Strategy Comparison

Figure 13: Comparison of sampling approaches.

that introduced by sampling only 2 times, and the broad practice for prior works, we did not choose sampling twice in the main results and keep it at 8.

Should we spread the budget across multiple critical positions? Instead of concentrating all samples at the top-1 position, we evaluate spreading the budget across the first 5 critical positions (3, 2, 1, 1, 1 samples respectively). As shown in Figure 13, this consistently degrades performance. The reason is structural to majority voting under a fixed budget: concentrating 8 samples at a single position yields a well-sampled estimate of that position's conditional distribution, and majority voting reliably amplifies its dominant mode. Spreading to 5 positions reduces each to only 1–3 samples, making per-position majority voting noisy, and cross-position vote mixing aggregates distributions that may disagree, washing out the signal. This is consistent with the MIT validation finding (§7) that MIT's top-1, while not always the absolute optimum, is reliably a high-effectiveness position — concentrating the budget there exploits MIT's ordinal precision without requiring pinpoint accuracy.

A.11 Propagation Kernels

A.11.1 Theoretical Analysis

To apply the principle of propagation, we model the reasoning trace as a directed graph and propose that the total, propagated influence can be measured by a diffusion process on this graph. Formally, this leads us to define a family of *diffusion kernels*, or equivalently *propagation operators*, parameterized by different matrix functions of the attention matrix $A \in \mathbb{R}^{N\times N}$. Each kernel describes how information diffuses across multiple hops, ranging from purely local propagation to global mixing, with different mathematical properties and constraints.

First Order Kernel The most basic form is the first-order kernel, which corresponds to one-step diffusion:

$$K_{\text{1-hop}} = A. \tag{14}$$

This operator captures only direct interactions between tokens, reflecting immediate influence between neighbors. While computationally efficient and stable, it cannot capture multi-hop dependencies.

Second Order Kernel To extend beyond immediate neighbors, we may incorporate two-step diffusion via the second-order kernel:

$$K_{\text{2-hop}} = A + A^2. \tag{15}$$

Here the term A^2 encodes influence transmitted through a single intermediary, while the additive form balances first- and second-order contributions. This kernel thus expands the local receptive field, but still truncates higher-order paths.

Resolvent Diffusion Kernel To model paths of arbitrary length, we consider the *resolvent diffusion kernel*, also known as the Green's function kernel, given by the Neumann series:

$$K_{\text{resolvent}} = \sum_{k=1}^{\infty} A^k = A(I-A)^{-1}, \tag{16}$$

which converges when the spectral radius $\rho(A) < 1$. This constraint ensures stability but may be too restrictive in practice. A damping parameter $\alpha \in (0,1)$ can be introduced:

$$K_{\text{resolvent},\alpha} = \sum_{k=1}^{\infty} (\alpha A)^k = \alpha A(I-\alpha A)^{-1}, \tag{17}$$

which guarantees convergence for all $\alpha < 1/\rho(A)$. The resolvent kernel assigns equal weight to paths of all lengths (modulo damping), allowing very long walks to contribute substantially. While expressive, this sensitivity to the spectrum of A and to the choice of α must be carefully controlled.

Heat Kernel An alternative diffusion kernel is the *heat kernel*, defined via the matrix exponential with a damping factor $\gamma > 0$:

$$K_{\text{heat}}(\gamma) = \exp(\gamma A) = I + \gamma A + \frac{(\gamma A)^2}{2!} + \frac{(\gamma A)^3}{3!} + \cdots. \tag{18}$$

Unlike the resolvent kernel, the heat kernel includes paths of all lengths but discounts them by $\gamma^k/k!$, strongly suppressing long walks. This factorial decay emphasizes short- and medium-range interactions. Moreover, the exponential series converges unconditionally for any square matrix A, making it numerically stable and well-defined regardless of spectral radius.

The spectral interpretation highlights these differences. If $A = Q\Lambda Q^{-1}$ with eigenvalues $\{\lambda_i\}$, then

$$K_{\text{heat}}(\gamma) = Q\,\mathrm{diag}(e^{\gamma\lambda_1}, \ldots, e^{\gamma\lambda_N})Q^{-1}. \tag{19}$$

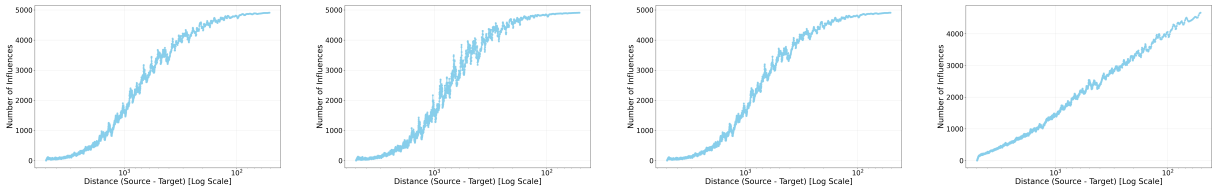
Thus the damping parameter γ acts as a nonlinear spectral filter: small values yield a local approximation close to $I+\gamma A$, while larger values amplify the contributions of dominant eigenmodes, enhancing global propagation. In contrast, the resolvent kernel corresponds to a rational spectral filter:

$$K_{\text{resolvent},\alpha} = Q\,\mathrm{diag}\left(\frac{\alpha\lambda_1}{1-\alpha\lambda_1}, \ldots, \frac{\alpha\lambda_N}{1-\alpha\lambda_N}\right)Q^{-1}, \tag{20}$$

which assigns equal geometric weight to paths of different lengths, modulated only by damping.

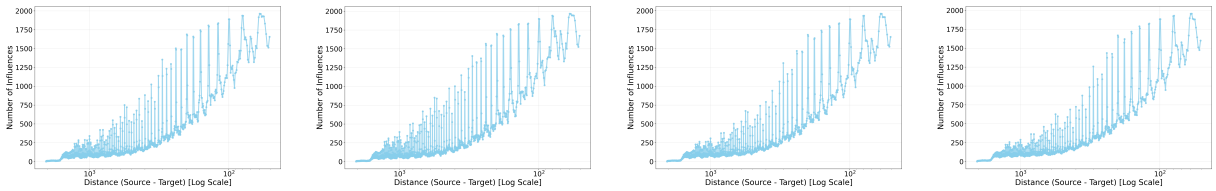
A.11.2 Empirical Analysis

We also conducted empirical comparison to complement our theoretical analysis, studying the impact of different kernels on both the reasoning model (DeepSeek-Distilled-Qwen-1.5B) and the non-reasoning model (LLaMA-3.1-8B) on the problem gpqa-diamond–0020. We first identify the most influential token for each token (target token) in the propagated attention matrix, then group them by the distance between the most influential token and the target token. This helps us evaluate the diffusion process spread across the graph. Overall we observe similar trends between different kernels for both model types.



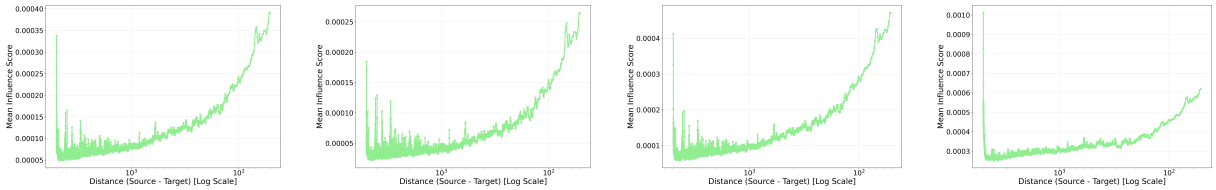
(a) Heat Kernel (b) First Order Kernel (c) Second Order Kernel (d) Resolvent Kernel

Figure 14: Influential Count Study for Propagation Kernels (**Reasoning Model**). This plot shows the distribution of distances between the most influential token and the corresponding target token.



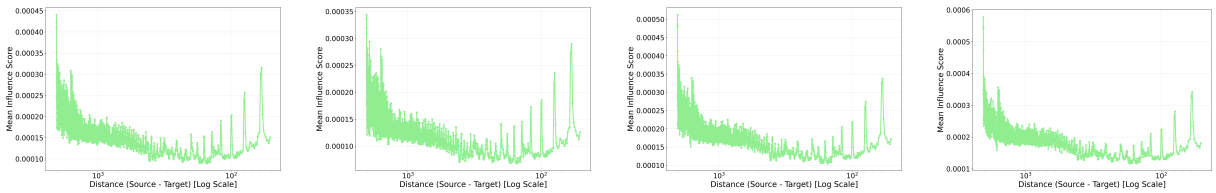
(a) Heat Kernel (b) First Order Kernel (c) Second Order Kernel (d) Resolvent Kernel

Figure 15: Influential Count Study for Propagation Kernels (**Non-Reasoning Model**). All kernels follow a similar trend. We also observe for LLaMA there are occasional spikes repeatedly occurring throughout the distance range.



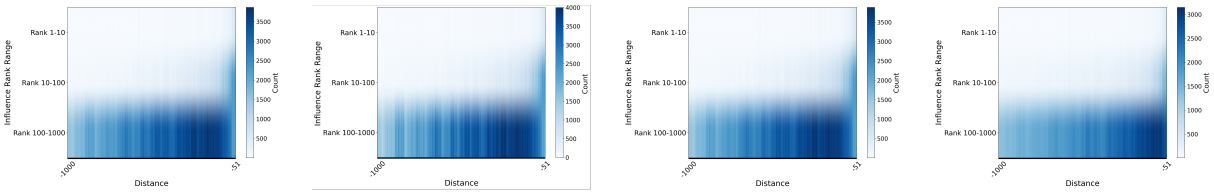
(a) Heat Kernel (b) First Order Kernel (c) Second Order Kernel (d) Resolvent Kernel

Figure 16: Mean Score Study for Propagation Kernels (**Reasoning Model**). This plot shows the distribution of influential score based on distances. We observe a spike in score when the distance is very far. We hypothesize that this resembles the attention sink pattern.



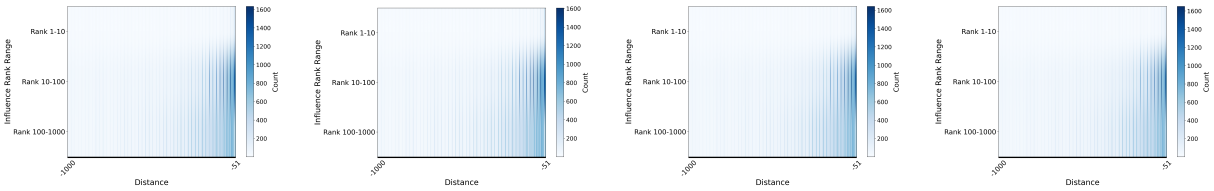
(a) Heat Kernel (b) First Order Kernel (c) Second Order Kernel (d) Resolvent Kernel

Figure 17: Mean Score Study for Propagation Kernels (**Non-Reasoning Model**). We notice a U shape in all kernels, where the mean score first goes down as distance increases, but goes up when the distance is very large. However such a pattern is not observed in the Reasoning Models.



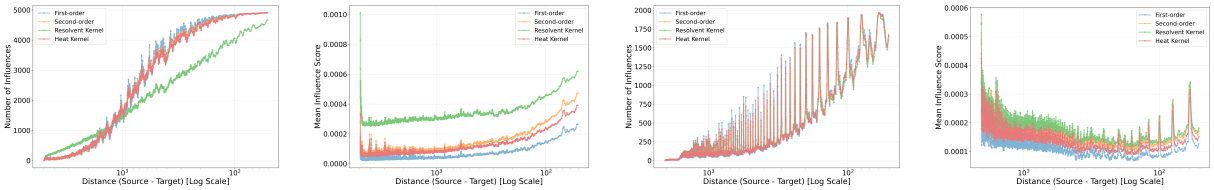
(a) Heat Kernel (b) First Order Kernel (c) Second Order Kernel (d) Resolvent Kernel

Figure 18: Rank Distribution Study for Propagation Kernels (**Reasoning Model**). For Resolvent Kernel where paths of different distances are considered equally, we observe a blurring effect where details are lost and hypothesize that it is the cause of the relatively low performance compared to the Heat Kernel.



(a) Heat Kernel (b) First Order Kernel (c) Second Order Kernel (d) Resolvent Kernel

Figure 19: Rank Distribution Study for Propagation Kernels (**Non-Reasoning Model**). All kernels exhibit similar patterns. The blurring effect in the non-reasoning model is not as strong as that of the reasoning model in Figure 18.



(a) Influential Count (**Reasoning**) (b) Influential Score (**Reasoning**) (c) Influential Count (**Non-Reasoning**) (d) Influential Score (**Non-Reasoning**)

Figure 20: Influential Count and Score Comparison across kernels. We observe similar trends in both model analyses

A.12 Prompts

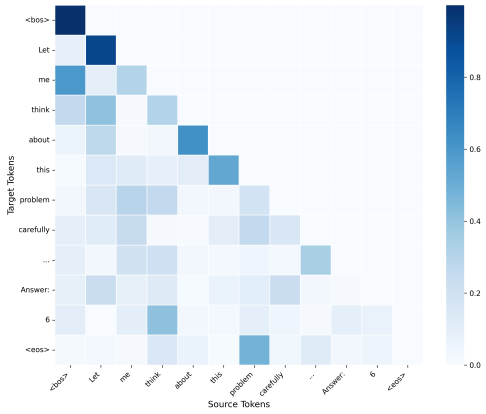
Below are the prompts used in testing. We have followed the prompts used in (Liu et al., 2025). And for the non-reasoning models like Gemma, we adopt the LLaMA prompt in that paper. For models with reasoning capabilities, we adopt the prompt of Qwen in that paper.

Model	Prompt
LLaMA-3.1-8B	Solve the following problem efficiently and clearly. - For simple problems (2 steps or fewer): Provide a concise solution with minimal explanation. - For complex problems (3 steps or more): Use this step-by-step format: ## Step 1: [Concise description] [Brief explanation and calculations] ## Step 2: [Concise description] [Brief explanation and calculations] ... Regardless of the approach, always conclude with: Therefore, the final answer is: `\boxed{answer}`. I hope it is correct. Where [answer] is just the final number or expression that solves the problem.
DeepSeek-Distilled–Qwen-1.5B	Please reason step by step, and put your final answer within `\boxed{}`.
SmolLM3-3B	Please reason step by step, and put your final answer within `\boxed{}`.
SmolLM3-3B(Think)	Please reason step by step, and put your final answer within `\boxed{}`.
Gemma-3-270M-IT	Solve the following problem efficiently and clearly. - For simple problems (2 steps or fewer): Provide a concise solution with minimal explanation. - For complex problems (3 steps or more): Use this step-by-step format: ## Step 1: [Concise description] [Brief explanation and calculations] ## Step 2: [Concise description] [Brief explanation and calculations] ... Regardless of the approach, always conclude with: Therefore, the final answer is: `\boxed{answer}`. I hope it is correct. Where [answer] is just the final number or expression that solves the problem.

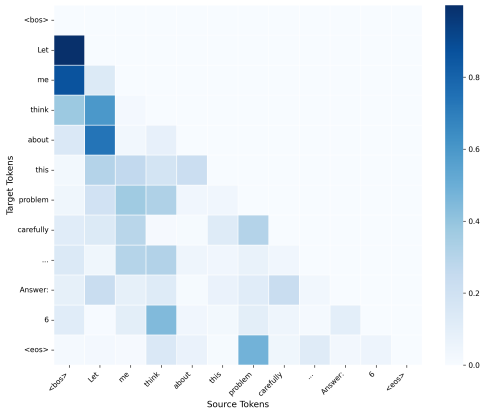
Table 5: Model Prompts

A.13 Examples

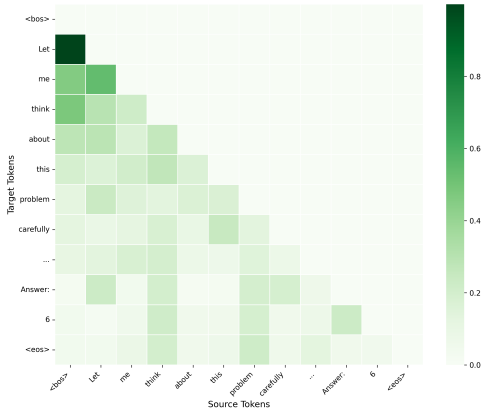
A.13.1 Example for MIT



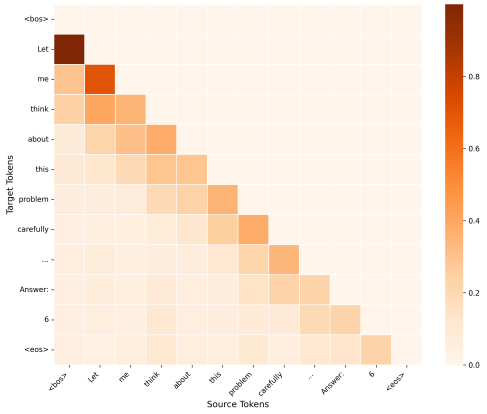
(a) Stage 0: Raw Attention (Layer 6, Head 3)



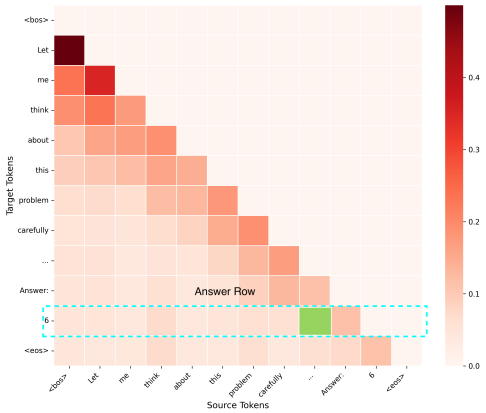
(b) Stage 1: Signal Purification (Layer 6, Head 3)



(c) Stage 2: Layer Aggregation (Head 3)



(d) Stage 3: Kurtosis-Weighted Head Aggregation



(e) Stage 4: Influence Propagation

Figure 21: Illustration example of the MIT algorithm. Raw attention goes through purification, layer aggregation, head aggregation and influence propagation to form the final score matrix. Only the rows related to the final answers are taken into consideration to select the most critical position.

A.13.2 Example for ACS with Reasoning Model

Question:

Given the sequence of words: apple shark vulture yak xerus. Concatenate the last letter of each word to form a new string. What is the resulting string?
Answer Choices:
(A) tkeks
(B) eteks
(C) ekeks
(D) ekejs
Please reason step by step, and put your final answer within \\boxed{}.

Original Answer Prefix

Let me try to think step by step. Okay, so I have this sequence of words: apple, shark, vulture, yak, xerous.

Wait

Original Answer Suffix

Wait, no, hold on, the sequence is apple, shark, vulture, yak, xerus. Wait, no, hold on, the sequence is apple, shark, vulture, yak, xerous. Wait, no, the user wrote: apple shark vulture yak xerus. So the words are: apple, shark, vulture, yak, xerous. Wait, hold on, xerous is a word? Or is it xerous? Wait, no, xerous is a word. Or is it xerous? Wait, no, xerous is a word. Wait, the last letters: apple ends with p, shark ends with k, vulture ends with t, yak ends with k, xerous ends with s. So the resulting string would be pktks.\n\nWait, but the options are (A) tkeks, (B) eteks, (C) ekeks, (D) ekejs. Hmm, none of these options match pktks. So maybe I misread the words.
Wait, let me check: the...Apple: last letter is t.\n\nShark: last letter is k.\n\nVulture: last letter is e.
Yak: last letter is k. Xerous: last letter is s.
So concatenated: tkeks. The final answer is: **\\boxed{A}**.

ACS Generated Suffix 1

I need to take the last letter of each word and then concatenate them all together to form a new string. Then, I have to figure out which of the given options is the resulting string. The options are (A) tkeks, (B) eteks, (C) ekeks, (D) ekejs. First, I need to make sure I understand the problem correctly. I have a list of five words: apple, shark, vulture, yak, xerous. For each of these words, I need to pick the last letter. Once I have those last letters, I concatenate them in the order they appear in the list to form a new string. Then, I have to see which of the given options matches this new string. Alright, so let's break it down step by step. Starting with the first word: apple. The last letter is 'e', right? Because 'apple' is spelled a-p-p-l-e. So the last letter is 'e'. Next, the second word is 'shark'. Let's see: s-h-a-r-k. So the last letter is 'k'....Concatenating these last letters in order gives us the string \"ekeks\".Thus, the resulting string is **\\(\\boxed{C}\\)**

ACS Generated Suffix 2

The task is to take the last letter of each word and concatenate them to form a new string. Then, among the given options, pick the correct one. The options are (A) tkeks, (B) eteks, (C) ekeks, (D) ekejs.Alright, let's break it down step by step. First, I need to identify the last letter of each word in the given sequence. The words are: apple, shark, vulture, yak, xerous.Starting with the first word, \"apple.\" The last letter is 'e.' So, the first character of the resulting string is 'e.'Next, the second word is \"shark.\" The last letter is 'k.' So, the second character is 'k.'\n\nMoving on to the third word, \"vulture.\" The last letter is 'e.' So, the third character is 'e.'Fourth word is \"yak.\" The last letter is 'k.' So, the fourth character is 'k.'Lastly, the fifth word is \"xerous.\" The last letter is 's.' So, the fifth character is 's.'Putting all these together, the resulting string is 'ekeks.'Looking at the options, (A) tkeks, (B) eteks, (C) ekeks, (D) ekejs. Hmm, 'ekeks' doesn't exactly match any of these. Wait, did I do that right?Let me double-check each last letter. Apple ends with 'e,' shark ends with 'k,' vulture ends with 'e,' yak ends with 'k,' ...Concatenating these last letters in order, we get: e, k, e, k, s, which forms the string \"ekeks\".Thus, the resulting string is **\\(\\boxed{C}\\)**

Figure 22: Example of how ACS allows correction for the generated answer. The critical position is the word *"wait"* in this case, allowing the following trace to ignore the hesitation on the sequence.

A.13.3 Example for ACS with Non-Reasoning Model

Question:

Name reactions in chemistry refer to a specific set of well-known chemical reactions that are typically named after their discoverers or the scientists who made significant contributions to their development. These reactions have had a profound impact on the field of chemistry and are often used as fundamental building blocks in various chemical syntheses. Identify the reactants for the following name reactions.
A + H2SO4 ---> 2,8-dimethylspiro[4.5]decan-6-one
B + BuLi + H+ ---> 4-methyl-1-phenylpent-3-en-1-ol

Answer Choices:
(A) A = 2,7-dimethyloctahydronaphthalene-4a,8a-diol, B = 4-methyl-1-phenylpent-3-en-1-one
(B) A = 2,8-dimethylspiro[4.5]decan-6-ol, B = 4-methyl-1-phenylpent-3-en-1-one
(C) A = 2,8-dimethylspiro[4.5]decan-6-ol, B = (((3-methylbut-2-en-1-yl)oxy)methyl)benzene
(D) A = 2,7-dimethyloctahydronaphthalene-4a,8a-diol, B = (((3-methylbut-2-en-1-yl)oxy)methyl)benzene

---

Original Answer Prefix

Step 1: To solve this problem, we first need to identify the name reactions and their corresponding reactants.
Step 2: The first reaction is A + H2SO4 ---> 2,8-dimethylspiro[4.5]decan-6-one. This reaction is likely the 2,7-dimethyloctahydronaphthalene-4a,8a-diol hydrolysis reaction.
Step

Original Answer Suffix

Step 3: The second reaction is B + BuLi + H+ ---> 4-methyl-1-phenylpent-3-en-1-ol. This reaction is likely the Friedel-Crafts alkylation of 4-methyl-1-phenylpent-3-en-1-one.
Step 4: Therefore, we can identify the reactants as A = 2,7-dimethyloctahydronaphthalene-4a,8a-diol and B = 4-methyl-1-phenylpent-3-en-1-one.
The final answer is: **A**

ACS Generated Suffix 1

Step 3: The second reaction is B + BuLi + H+ ---> 4-methyl-1-phenylpent-3-en-1-ol. This reaction is likely the 4-methyl-1-phenylpent-3-en-1-one alkylation reaction, but it seems like it's a lithiation reaction of the benzene ring,.. The given reactant for this reaction is likely (((3-methylbut-2-en-1-yl)oxy)methyl)benzene.
Step 4: Based on the above analysis, we can conclude that the correct reactants for the given name reactions are A = 2,7-dimethyloctahydronaphthalene-4a,8a-diol and B = (((3-methylbut-2-en-1-yl)oxy)methyl)benzene.
Step 5: Therefore, the correct answer is option D.
The final answer is: **D**

ACS Generated Suffix 2

Step 3: The second reaction is B + BuLi + H+ ---> 4-methyl-1-phenylpent-3-en-1-ol. This reaction is likely the Grignard reaction. In a Grignard reaction, a Grignard reagent (RMgX) reacts with a carbonyl compound to form an alcohol. The Grignard reagent is formed by the reaction of an alkyl halide with magnesium.
Step 4: The Grignard reagent formed in this reaction is likely 4-methyl-1-phenylpent-3-en-1-ylmagnesium bromide, which reacts with H+ to form the alcohol 4-methyl-1-phenylp...which would then react with 4-methyl-1-phenylpent-3-en-1-one to form the alcohol 4-methyl-1-phenylpent-3-en-1-ol.
Step 5: Therefore, the reactants for the given name reactions are A = 2,7-dimethyloctahydronaphthalene-4a,8a-diol and B = (((3-methylbut-2-en-1-yl)oxy)methyl)benzene.
The correct answer is (D).
Step 6: Therefore, the final answer is (D).
Step 7: Therefore, the final answer is **D**

Figure 23: Example of how ACS allows correction for the generated answer. The critical position is the word *"step"* in this case. A different step 3 leads to the correct answer.

A.13.4 Failure Example for ACS

Question:
Given the sequence of words: turtle carrot hippo. Concatenate the last letter of each word to form a new string. What is the resulting string?

Answer Choices:
(A) eho
(B) etc
(C) eto
(D) etv

Original Answer Prefix

Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n ...Final Answer: (A)\nFinal Answer: (A)\nFinal

Answer

Original Answer Suffix

Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n ...Final Answer: (A)\n Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n ...Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n
Final Answer: **A**

ACS Generated Suffix 1

Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n ...Final Answer: (A)\n Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n ...Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n
Final Answer: **A**

ACS Generated Suffix 2

Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n ...Final Answer: (A)\n Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n ...Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n
Final Answer: **A**

Figure 24: Example of how ACS fails. A lot of the time the failure comes from identifying a critical position too late, where the model is already committed to an answer.

A.14 Potential Impact and Future Works

Our work demonstrates that LLM reasoning is not a uniform chain of steps but a process punctuated by critical positions. We define such critical positions as locations where cognitive or computational resources would make the most impact. We introduced **Metacognitive Influence Tracing (`MIT`)**, a principled diagnostic method to identify these positions, and showed how this understanding can be translated into a high-efficiency interventive reasoning framework, **Adaptive Critical Sampling (`ACS`)**. The ability to pinpoint these critical moments has implications that extend beyond improving Chain-of-Thought prompting, touching on challenges in AI safety, evaluation, and future model development.

Implications for AI Safety and Alignment. A key challenge in AI safety is understanding and preventing undesirable model behavior, such as hallucinations or harmful content. Our results, similar to findings on "forking tokens," (Bigelow et al., 2025) suggest that a model may be just one critical step away from producing a radically different—and potentially unsafe—output. `MIT` can serve as a diagnostic tool for failure analysis: when a model produces an undesirable result, `MIT` traces the reasoning back to the specific flawed inference that initiated the failure. This moves beyond detecting that an output is wrong, providing mechanistic insight into *where* it could be corrected, and enabling targeted safety interventions.

Implications for Model Development and Evaluation. The insights from `MIT` could inform more efficient model design. By aggregating critical positions across thousands of problems, we can potentially map the model components (e.g., attention heads or layers) most crucial for reasoning. This data could guide structured pruning or distillation, yielding smaller models that preserve disproportionate reasoning capability. Furthermore, our findings suggest that static evaluations measuring only the final answer can be misleading. A model might appear confident in its conclusion, yet `MIT` reveals that it passed through highly uncertain positions where outcomes could have easily diverged, exposing brittleness hidden by the final result.

A promising direction is to integrate `MIT` with other reasoning frameworks. Critical positions identified by `MIT` could guide the branching of search algorithms such as Tree of Thoughts, concentrating expansion at points of highest leverage. This would combine the breadth of exploration with the precision of targeted intervention, paving the way for hybrid reasoning systems. Applying `ACS` repeatedly can also enable continuous improvement.

Another future direction is to zoom out the scope: instead of using tokens as the basic unit, we can zoom out to sentence or paragraph level as the basic unit of thought. This can provide another granularity on the reasoning process of LLMs.

A.15 Use of LLMs

We have used LLMs to polish writing for this paper.