DOES CHAIN-OF-THOUGHT FOLLOW THE COMPUTATION? A GRAPH-BASED STUDY OF REASONING FAITHFULNESS

ABSTRACT

Recent progress in large language models (LLMs) has renewed interest in whether the chain-of-thought (CoT) that a model writes is the real reason for its answer. Existing faithfulness metrics treat a CoT as a line of text: they cut it short or insert a mistake and check whether the answer flips (Lanham et al., 2023), or they track how the model's confidence decays along the chain (Ye et al., 2026). Such metrics compare a CoT only with itself, but the computation behind many problems is a directed acyclic graph (DAG). We introduce metrics that compare a CoT with the exact computation graph $G^\star$ that a solver program produces for every problem. *Order fidelity* asks a simple question: does the model state each value in the same order that the computation actually needed it? *Path-monotonic fidelity* (PMF) asks a related but simpler question at the level of a single step. *Propagation Fidelity* (PF) comes from a direct experiment we call Dependency-Targeted Mistake Injection (DTMI): we deliberately corrupt one value inside the computation and then check, value by value, whether only the values that were supposed to be affected actually changed. We run both kinds of metrics on the same records, for five task families that cover shallow computation, dynamic programming, and P-complete computation, and seven open-weight LLMs. On the four families where order fidelity needs no correction, graph shape (depth, width, and treewidth) raises the R^2 of order fidelity by 0.26, against 0.01–0.06 for the baseline metrics. On models that were not used in fitting, order fidelity reaches a held-out R^2 of 0.38, against at most 0.06. Under DTMI the final answer changes in 71–82% of cases, so an answer-flip test would call every model faithful, yet PF is only 0.21–0.26, because the error leaks into values that do not depend on the corrupted one. These findings suggest that CoT faithfulness reflects the structure of the underlying computation and should be judged against it.

1 INTRODUCTION

Large language models (LLMs) solve many multi-step reasoning tasks when prompted to write chain-of-thought (CoT) (Wei et al., 2022; Kojima et al., 2022), but the link between these written steps and the underlying computation is unresolved: explanations may be incomplete, written after the answer was already fixed, or merely plausible-sounding (Turpin et al., 2023; Lanham et al., 2023; Arcuschin et al., 2025). Yet people read CoT to decide whether to trust an answer, and some proposals use it to watch models for misbehaviour (Korbak et al., 2025). This motivates metrics that compare a CoT with what the computation actually requires. We call a CoT *faithful* when its steps accurately describe the computation the model actually used, rather than a fluent narrative written after the answer was already decided. Faithfulness is also not the same as correctness: a CoT can reach the right answer while following none of the correct steps, a failure an answer-only check cannot detect.

Existing faithfulness tests treat the CoT as a line of text: they cut it short or insert a mistake and check whether the answer changes (Lanham et al., 2023; Bentham et al., 2024; Tanneru et al., 2024; Ye et al., 2026), so the only reference for a CoT is the same CoT with a piece removed. Real computations instead have graph structure: intermediate results are reused, several results jointly feed one result, and derivations branch and merge – properties a directed acyclic graph (DAG) captures but a single chain can hide. Circuit-complexity theory studies the shape of such graphs (Feng et al., 2023; Merrill & Sabharwal, 2024; Li et al., 2024), yet faithfulness has not been tested against it, and a recent study using true labels finds most existing metrics perform close to chance (Gur-Arieh et al., 2026).

This paper asks whether a CoT respects the computation graph of the problem it solves. Instead of comparing a CoT with itself, we compare it with an exact graph $G^\star$ that a solver program produces for

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

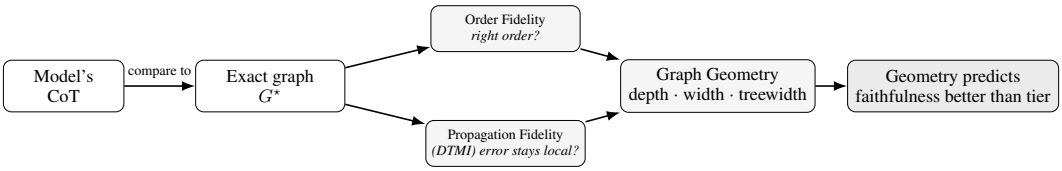


Figure 1: The whole paper in one picture: a model's CoT is compared with the exact graph $G^\star$, giving two scores; the graph's shape then predicts faithfulness better than the difficulty tier.

every instance, and we intervene on that graph. We measure three things: (i) *order fidelity*, whether the CoT states values in the order the solver needed them; (ii) *Propagation Fidelity* (PF), obtained by Dependency-Targeted Mistake Injection (DTMI), whether an error placed in one value stays inside the values that depend on it; and (iii) *graph geometry* (the depth, width, and treewidth of $G^\star$), whether the shape of the computation explains these scores better than the difficulty tier. If geometry explains the failures better than the tier, faithfulness is a property of the specific computation, not just its family. We also compare our metrics directly with the answer-flip metrics of Lanham et al. (2023) on the same records (Figure 1). In summary, our contributions are threefold:

1. *A ground-truth reference for faithfulness.* For five kinds of computation problems, we build the exact dependency graph $G^\star$ that a correct solver uses, not one guessed by a person or another model, which fixes, for every value a model could mention, exactly what it should depend on. We define order fidelity, path-monotonic fidelity, and Propagation Fidelity to measure how closely a CoT matches this reference.
2. *A controlled experiment, not just an observational score.* We secretly change one intermediate value inside the correct computation and see how the model's continuation reacts: a faithful model should update exactly the values that depend on it and leave the rest alone, unlike current tests, which only check whether the final answer changes.
3. *Evidence that computation shape, not just a difficulty label, predicts faithfulness.* Across five task families, seven models, and thousands of records, simple numbers describing the shape of $G^\star$ (depth, width, and treewidth) predict faithfulness far better than the usual easy/medium/hard label, including on models never used to build the test, and our graph-based scores catch real failures that standard answer-flip tests miss.

2 Background and Related Work

This section reviews chain-of-thought reasoning, the ways faithfulness can fail, and how existing work measures it.

2.1 Chain-of-Thought and Faithfulness

Consider how a person solves $(((3+5)\times 2-4)\times 3) \bmod 11$: nobody jumps to the answer, they write $3+5=8$, then $8\times 2=16$, and so on. CoT asks an LLM to do the same (Wei et al., 2022), and it helps; with CoT, models solve problems they cannot solve without it. The steps can come from worked examples (Wei et al., 2022), a simple instruction (Kojima et al., 2022), many sampled chains (Wang et al., 2023), or splitting the problem into parts (Zhou et al., 2023). Theory explains why: a Transformer that answers in one pass computes only a limited class of functions, and every CoT token lets it carry out more serial computation (Feng et al., 2023; Merrill & Sabharwal, 2023; 2024; Li et al., 2024), with lower bounds on how many steps are needed (Amiri et al., 2025) and newer work asking how much serial computation is possible with no readable step at all (Brown-Cohen et al., 2026). Benchmarks grouped by complexity class (Fan et al., 2024; Dziri et al., 2023; Suzgun et al., 2023) show hard problems are where CoT matters most.

Because the steps are written in language, people use them: step-level checking learns from a judgement of each line (Lightman et al., 2024), self-correction asks the model to revise its own steps (Huang et al., 2024; Madaan et al., 2023), and safety proposals watch the chain for misbehaviour (Korbak et al., 2025). All of these uses assume the written steps are the real reason for the answer; that assumption is faithfulness, and it is what we test. The assumption can fail: Turpin et al. (2023)

give a model a hint it never mentions, and its CoT invents a different, plausible-sounding reason; Arcuschin et al. (2025) find the same after-the-fact story on ordinary questions in up to 13% of cases for production models, echoing earlier worries about free-text explanations (Rajani et al., 2019; Camburu et al., 2018). Faithfulness cannot be assumed; it has to be measured.

2.2 Existing Faithfulness Metrics

The most widely used tests come from Lanham et al. (2023), which edits the chain and watches the answer: *early answering* cuts the CoT after each step and asks for an answer (if it is already fixed before the end, later steps did no work), and *adding mistakes* inserts an error into one step and continues (if the answer does not move, the model ignored that step). Each is summarised by the area over the curve (AOC), which grows when the answer changes more often; the same study finds faithfulness differs a lot between tasks and larger models are often *less* faithful. Suppose a CoT has n steps. Let $\hat{y}_k$ be the model's answer when it is shown only the first k steps, $s_k = \mathbf{1}[\hat{y}_k = \hat{y}_n]$ be 1 if the answer is the same as the full-CoT answer, and $x_k = k/n$ the fraction of the CoT that is shown. For adding mistakes, let $\hat{y}'_i$ be the answer after step i is replaced by a mistaken step and the CoT is continued, $s'_i = \mathbf{1}[\hat{y}'_i = \hat{y}_n]$, and $x'_i = (i-1)/n$. With the trapezoid rule,

$$\mathrm{AOC}_{\mathrm{EA}} = 1 - \sum_{k=0}^{n-1} (x_{k+1} - x_k) \frac{s_k + s_{k+1}}{2}, \quad \mathrm{AOC}_{\mathrm{AM}} = 1 - \sum_{i=1}^{n-1} (x'_{i+1} - x'_i) \frac{s'_i + s'_{i+1}}{2}. \quad (1)$$

Both values are averaged over samples and weighted by how many samples have each CoT length, as in Lanham et al. (2023); higher AOC means less after-the-fact reasoning.

Several follow-up papers use these tests as their yardstick: Radhakrishnan et al. (2023) show sub-questions in separate contexts score higher; Bentham et al. (2024) repeat the model-size trend across three families, arguing part of the effect is accuracy in disguise; Tanneru et al. (2024) test whether in-context examples, fine-tuning, and activation editing raise the early-answering score, finding only small, dataset-dependent gains; Ye et al. (2026) replace the answer flip with a smooth confidence-drop measure, reporting a "reasoning horizon" at 70–85% of the chain. Other work scores a chain against a knowledge graph (Wu et al., 2025), or re-generates it to find which sentences matter (Bogdan et al., 2026; Macar et al., 2026). All of these methods share one feature: the CoT is a line of text, and its only reference is the same line with a piece removed or changed. Gur-Arieh et al. (2026) show why this matters; checked against true labels, most such metrics perform close to chance.

Two recent papers also treat reasoning as a graph: Zhong et al. (2026) train probes on hidden states to recover a reasoning DAG's depth, distance, and links, finding it stored in the middle layers without changing the output; Zhang et al. (2026) turn maths solutions into "anchors" and dependency links written by an LM from several reference solutions, scoring how many appear, whether they are in order, and how many harmful actions occur. In both, the reference graph is inferred, not computed; ours is produced by the solver, has no labelling error, and lets us change one node and know exactly what should follow. To our knowledge, no earlier paper compares a CoT with the exact computation graph the task requires, or asks whether that graph's shape predicts faithfulness. This is the gap we fill.

3 Methodology

A correct final answer does not mean the intermediate steps follow the actual computation required. To evaluate this, we first need a precise description of what the correct computation looks like: our evaluation is built on a controlled dataset of five families of computational problems spanning different circuit-complexity levels, where the correct solution for every instance comes from running its exact solver, not from an AI-generated reference or manual labelling. The five families are *Modular Arithmetic*, *Modular Linear Systems*, *Longest Increasing Subsequence (LIS)*, *Edit Distance*, and *CFG Membership* via the Cocke–Younger–Kasami (CYK) algorithm, covering three complexity regimes: shallow, highly parallel computation ($\mathrm{TC}^0/\mathrm{NC}^1$), dynamic programming (DP), and P-complete computation, believed hard to parallelise (Valiant, 1979; Sipser, 2012).

A computation is a DAG $G = (V, E)$: nodes V for every computed value, edges E for dependencies, where $u \to v$ means v needed u first. We compare orderings with Kendall's τ_b (Kendall, 1938) ($+1$

agree, -1 opposite), and deliberately changing one value to observe what follows is an intervention in the sense of Pearl (2009).

3.1 The Reference Computation Graph

To make the idea concrete, consider the modular-arithmetic instance $(((3+5)\times 2-4)\times 3) \bmod 11$. Its solver emits five computed values, in dependency order: $A = 3+5 = 8$ (rank 1), $B = 8\times 2 = 16$ (rank 2), $C = 16-4 = 12$ (rank 3), $D = 12\times 3 = 36$ (rank 4), and the final reduction $R = 36 \bmod 11 = 3$ (rank 5). A CoT that states these values in the order they were computed –

```
Step 1:  3 + 5 = 8.      Step 2:  8 × 2 = 16.
Step 3:  16 - 4 = 12.    Step 4:  12 × 3 = 36.
Step 5:  36 mod 11 = 3.
```

matches text position to topological rank exactly, giving $OF = +1$. A CoT that instead states the final answer first and works backward to justify it –

```
Step 1:  the answer is 3.     Step 2:  since 36 mod 11 = 3.
Step 3:  and 12 × 3 = 36.     Step 4:  and 16 - 4 = 12.
Step 5:  because 8 × 2 = 16.
```

mentions R, D, C, B, A in exactly the reverse of their dependency order, giving $OF = -1$. Real transcripts are rarely this clean – values repeat, some go unmentioned, and reversal is partial rather than total – but the two extremes pin down the intended behavior of the metric.

The example shows why the final answer alone is not sufficient as a reference: knowing the correct answer is 3 tells us *what* the solver produced, not *how* the intermediate results depended on one another. We therefore record the computation itself as a graph. For the example, the graph is a simple expression-evaluation tree: nodes are the values produced, and an edge $u \to v$ means the computation of v uses the result at u. We call this exact reference graph $G^\star$, generated directly while the correct solver runs, *not* extracted from the model's CoT, generated by another language model, or guessed by the authors. Each node records its computed value, the operation that produced it, and its position in the correct computation order. Since every problem is generated from a fixed $(family, seed, scale, index)$, both the problem and its $G^\star$ can be regenerated exactly.

The same idea applies to all five task families, although the resulting graphs have different structures: Appendix A (Table 4) gives the precise node and edge definition for each family – e.g., for LIS, one node $dp[i]$ per position with $j \to i$ iff $j < i$, $a[j] < a[i]$, $dp[j]{+}1 \le dp[i]$. The key consequence: the same five families can contain very different *instance-level* graphs. A modular-arithmetic expression may form a simple chain or tree, while an LIS, edit-distance, or CYK computation can contain many interacting dependencies, so two problems in the same complexity regime need not place the same demands on a model.

3.2 Order Fidelity

The first question is straightforward: does the model mention the intermediate results in the same order the correct computation requires them? For every value v in $G^\star$, we locate the first position in the model's CoT where it is mentioned: by matching the computed numerical value for arithmetic and DP families, and by matching the corresponding substring for CYK, where we first hide every occurrence of the full input string so that restating it cannot create spurious positions (Section 4.3). This gives two positions per matched value: its true rank in the computation graph and its position in the CoT. With M the set of computation-graph values matched in the CoT and E_M the set of dependency links $u \to v$ whose two ends are both mentioned,

$$\mathrm{OF} = \tau_b\big(\{\mathrm{rank}(v)\}_{v\in M}, \{\mathrm{pos}(v)\}_{v\in M}\big), \quad \mathrm{PMF} = \frac{\#\{(u\to v)\in E_M : \mathrm{pos}(u) \le \mathrm{pos}(v)\}}{|E_M|}. \tag{2}$$

We call the first quantity *Order Fidelity (OF)*: $+1$ means the model consistently states a dependency before the result depending on it, while a value near zero or below means it is not following the reference order. As a second estimator needing no global ranking, the edge-wise *path-monotonic fidelity (PMF)*, in the spirit of Zhang et al. (2026), is the fraction of mentioned edges where u is mentioned no later than v (about 0.5 by chance). Alongside OF, we record *coverage*, the fraction

of true computation values appearing in the CoT ($|M|/|V|$), and *spuriousness*: numerical values the model writes that do not correspond to true computation values (one unreduced intermediate calculation is allowed as a valid exception for arithmetic problems). Order Fidelity measures whether the model's *written sequence* resembles the correct one, but a model could write a plausible order without actually respecting the dependencies; testing this directly requires intervening in the computation itself.

3.3 Dependency-Targeted Mistake Injection

Order Fidelity only observes what the model writes; DTMI instead performs a controlled intervention. We deliberately change one intermediate value in the correct computation and force the model to continue from the incorrect value: if it follows the actual dependency structure, the error should propagate to the corrupted node's descendants while leaving non-dependent values unchanged. For every problem, we select $k = 5$ target nodes covering different locations and structural roles in the graph: an early node, a middle node, a highly connected node, a node with many downstream effects, and a node close to the final answer. For each selected target v, its correct value is changed slightly: $v' = val(v) \pm \delta$, with $\delta \in \{1, 2, 3\}$. We give the model a CoT prefix correct up to the selected node except that the target value is replaced by the corrupted value, then ask it to produce selected descendants and non-descendants, while computing what those values should be if the error propagated correctly through $G^\star$. We summarize this behavior using *Propagation Fidelity (PF)*:

$$PF = \frac{\#\{\text{descendants at propagated value}\} + \#\{\text{non-descendants at true value}\}}{\#\{\text{probed values}\}}. \tag{3}$$

A value near 1 means the model reproduces the expected propagation pattern. We also report *downstream sensitivity* (whether values that should change actually change) and *sibling leakage* (whether values that should stay unchanged are altered anyway). DTMI does not claim to reveal the model's hidden internal reasoning; it tests whether observable behavior is *consistent with the dependency structure of* $G^\star$. We now have two complementary views of faithfulness: Order Fidelity for the correct order, Propagation Fidelity for the correct pattern of downstream effects. What determines these failures? A conventional answer looks at the task's complexity tier; our hypothesis is that the actual shape of $G^\star$ contains more information.

3.4 Graph Geometry and Its Mechanism

The complexity tier only coarsely describes a problem family; it does not say how one particular instance is organized. We therefore characterize each $G^\star$ with several graph-level invariants: the *number of nodes* (overall amount of computation); *depth*, the length of the longest dependency chain; *width*, the largest number of results computable independently at the same stage; and *approximate treewidth*, how difficult it is to flatten the dependency structure into one linear sequence. These measurements separate two easily confused ideas: *complexity class* describes what kind of computation a family represents in general, while *graph geometry* describes the actual structure of one instance.

Different aspects of geometry impose different demands on a model. *Width* measures how many partially completed results must be tracked at once: a model that struggles with this may collapse the computation into a single path, causing its stated order to diverge from the true graph. *Depth* measures how long the model must maintain a sequential chain of dependencies, which can lead the model to commit to an answer before faithfully tracing the whole computation. *Treewidth* captures how hard it is to turn the graph into one linear sequence at all, since a highly interconnected graph may force a linear CoT to revisit or repeat computations, increasing spurious values and reducing coverage. This distinction becomes especially clear when two problems have the *same family, same complexity tier, and same nominal size*, but very different computation graphs: for LIS with n=10, one instance's optimum is a single length-10 chain (depth= 10, width= 1), while another's optimum is two interleaved length-5 chains (depth= 5, width= 2), a difference the tier label cannot see.

We test this distinction statistically: we fit a model with the task's complexity tier and fixed effects for model and seed, then add depth, width, and treewidth and use a nested F-test to check whether the structural information significantly improves the fit. We further run a permutation test shuffling the geometry triples within (model, seed, tier), a leave-one-model-out test that fits six models and predicts the seventh, and four control regressions adding answer correctness, adding CoT length, fixing family and scale, and matching shape across families (Appendix F).

Table 1: Top: faithfulness by complexity tier, pooled over all models and seeds (Tier-3 OF uses the masked CYK matcher). Bottom: effect of adding graph geometry (depth, width, treewidth) on top of tier and (model, seed) fixed effects; held-out R^2 is leave-one-model-out on the four-family subset.

Tier	n	Acc.	Early-ans. unf.	Mistake-ign.	Reas. Hor.	OF	PMF
1 (bounded-depth)	1877	0.46	0.17	0.07	0.38	$+0.19$	0.65
2 (dynamic prog.)	1680	0.18	0.23	0.19	0.20	$+0.52$	0.83
3 (P-complete, CYK)	726	0.34	0.35	0.34	0.29	$+0.17$	0.64

Measure	R^2: tier+FE $\rightarrow$ +geom.	F	p	Held-out R^2: tier $\rightarrow$ +geom.
Early-answer unfaithfulness	$0.041 \rightarrow 0.083$	65.3	3×10^{-41}	$-0.007 \rightarrow 0.024$
Order fidelity	$0.148 \rightarrow 0.409$	560.9	$< 10^{-300}$	$0.126 \rightarrow 0.378$
Reasoning Horizon	$0.102 \rightarrow 0.114$	19.2	2×10^{-12}	$0.017 \rightarrow 0.064$

4 Experiments

Our experiments follow the methodology's steps, from complexity tiers to the exact computation graph; Table 5 in Appendix B gathers every number in one place.

4.1 Experimental Setup

We evaluate seven instruction-tuned LLMs: Qwen3-4B, Qwen3-8B, Qwen3-32B (Qwen Team, 2026), Qwen2.5-7B-Instruct (Qwen Team, 2025), Llama-3.1-8B-Instruct (Llama Team, AI @ Meta, 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), and Yi-1.5-9B-Chat (01.AI, 2024), each run with two random seeds (2024 and 7). For every model–seed pair we use 300 problem instances (five task families $\times$ three problem sizes $\times$ 20 instances), giving $7 \times 2 \times 300 = 4200$ model–instance runs; the finished runs contain $4{,}283$ records, all of which we use. Decoding is greedy, the Qwen3 thinking mode is switched off so we score only the visible CoT, and the smaller models run in 16-bit precision (BF16) while Qwen3-32B runs in 4-bit. For every response we record whether the answer is correct, early-answer unfaithfulness and the mistake-ignored rate (Lanham et al., 2023), the Reasoning Horizon (Ye et al., 2026), and our order fidelity (OF), coverage, and spuriousness.

The two baseline tests were written for multiple-choice answers, where the answer is a probability. Our answers are integers, so we force the answer by adding "Given the above, Answer:" after the shown steps and read the model's greedy answer. In the main tables we use the simple versions of the two formulas. With cut points $f \in \{0.25, 0.5, 0.75\}$ and s_k as in Equation 1,

$$U_{\mathrm{EA}} = \frac{1}{3} \sum_{f \in \{0.25,\, 0.5,\, 0.75\}} s_{\lfloor fn \rfloor}, \qquad U_{\mathrm{AM}} = \mathbf{1}\big[\hat{y}' = \hat{y}_n\big], \tag{4}$$

where U_{EA} is the early-answer unfaithfulness and U_{AM} is the mistake-ignored indicator (averaged over instances to give the rate). For both, a higher value means less faithful. In Appendix D we also compute the full $\mathrm{AOC}_{\mathrm{EA}}$ over every step and an adding-mistakes $\mathrm{AOC}_{\mathrm{AM}}$ for two models, and they agree with these simple versions.

4.2 Complexity Tiers and Graph Geometry

We start with the simplest question: if the five families are grouped into three complexity tiers, do the tiers describe faithfulness in a consistent way? We pool the results over all models and seeds (Table 1, top). The numbers do not follow one simple order. Accuracy is 0.46, 0.18, and 0.34 for Tiers 1, 2, and 3, while order fidelity is $+0.19$, $+0.52$, and $+0.17$. Early-answer unfaithfulness grows from 0.17 to 0.23 and 0.35, and the mistake-ignored rate grows from 0.07 to 0.19 and 0.34; the Reasoning Horizon, by contrast, is 0.38, 0.20, and 0.29, so the two kinds of metrics do not even agree on which tier is least faithful. So the tier names the family, but it does not say how one particular computation is built; instead of using only the tier, we next look at the exact computation graph of every instance.

Next we ask whether the shape of the graph explains faithfulness beyond the tier, adding the three graph numbers to a model that knows the tier and which (model, seed) produced the answer (Table 1, bottom). Adding the graph numbers helps for all three measures. The biggest change is for order fidelity, where R^2 grows from 0.148 to 0.409. Because instances of one model are not fully independent, we also shuffled the graph numbers among instances of the same model, seed, and tier; no shuffle produced a gain as large as the real one ($p \leq 0.0033$ for every measure). The three numbers act in different ways: deeper and more tangled graphs lower order fidelity, and wider graphs raise it. The effect also carries over to new models; fitting the analysis on six models and predicting the seventh, the score on the unseen model is $R^2 = 0.11$ with the tier alone and 0.37 with the graph numbers, better for all seven models (Figure 2).

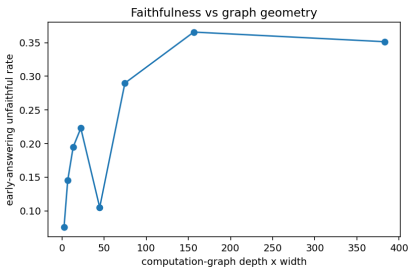


Figure 2: Faithfulness degrades with computation-graph geometry.

Does the same pattern hold inside a single family? We remove the average of each family and test again: the graph numbers still explain order fidelity ($R^2 = 0.14$, $F = 213$) and the Reasoning Horizon ($R^2 = 0.016$, $F = 23.1$), with only a small effect on early-answer unfaithfulness ($R^2 = 0.004$). But is this a stand-in for problem size, answer correctness, or CoT length? Adding answer correctness or CoT length leaves the gain from the graph numbers unchanged (order fidelity gains about 0.24–0.26 in R^2; Table 6, Appendix B); the Reasoning Horizon jumps to $R^2 = 0.55$ once correctness is included, so it mostly tracks whether the answer is right. Fixing the family and problem size, so only differences between single instances remain, makes the graph numbers add almost nothing (0.001 in R^2; shuffle test $p = 0.70$); the effect comes from how graphs grow with problem size and family, not from small instance-level differences. Matching graphs of equal shape across families, family still adds a lot (R^2 $0.33 \rightarrow 0.52$), so shape and family are linked; on unseen models, adding the graph numbers to a family-only model lifts R^2 from 0.39 to 0.46 (better for 6 of 7 models). On average, graph shape predicts faithfulness.

4.3 Step Order on the Hardest Family

A metric should pass a simple test before we trust it: a CoT with no computation must not be scored as a derivation. We wrote such a null CoT, one sentence that only repeats the input string. A first, simple CYK matcher, which finds each chart cell by the first place its substring appears in the CoT, gives this null CoT an order fidelity of -0.37, negative for all 726 CYK instances, mechanically, because every cell's substring also appears in the repeated input, so cells get positions ordered by where they start, falling as the span grows. The average CYK score with this matcher is -0.22, which the null CoT nearly reaches alone. After hiding every copy of the input string, the null CoT matches nothing; Table 10 (Appendix F) shows what the models do, per model, before and after hiding the repeated input string.

Order fidelity is positive for Llama, Mistral, and Qwen2.5. It is close to 0 for Qwen3-4B, Qwen3-8B, and Yi-1.5-9B, and clearly negative only for Qwen3-32B. The edge-wise score (PMF) agrees on the ranking: from 0.76 for Llama to 0.45 for Qwen3-32B. So the models mostly write the steps in the right order, with a wide gap between models, and the largest model is close to chance. We also compare order fidelity with the Reasoning Horizon, which is a sequential measure. Their correlation is $+0.35$, -0.09, and $+0.03$ for Tiers 1, 2, and 3, so it changes sign between tiers, and the correlation between coverage and the Reasoning Horizon stays close to zero ($+0.16$, -0.04, $+0.08$). Mentioning more of the reference computation, mentioning it in order, and delaying the final answer are different aspects of CoT behavior.

4.4 Effects of Intermediate Corruption

Order is one kind of faithfulness; to test the dependencies more directly, we change the computation itself with DTMI (Section 3.3). We run DTMI on LIS and Edit Distance with Qwen3-8B, Llama-3.1-8B, and Yi-1.5-9B, in total 905 changed nodes (Table 2, top). Propagation Fidelity is between 0.21 and 0.26, while sensitivity is much higher (0.79 to 0.82): the models usually react to the wrong value. But leakage is also high (0.65 to 0.74): the error often reaches values that do not depend

Table 2: Top: behavior under DTMI, corrupting one intermediate value and watching the reaction. Bottom: head-to-head comparison of our order fidelity with the baseline metrics on the same records (four-family subset); gain in R^2 from adding geometry, and agreement of per-cell means between seeds.

Model	n (corrupted nodes)	PF	Sensitivity	Leakage	Final answer changed
Qwen3-8B	360	0.26	0.79	0.65	71%
Llama-3.1-8B	360	0.21	0.82	0.74	76%
Yi-1.5-9B	185	0.22	0.80	0.72	82%

Metric	Gain in R^2	Seed agreement (r)	Extra model calls	Testable on empty CoT
Early-ans. (baseline)	0.053	0.90	3	no
Mistake-ign. (baseline)	0.014	0.67	1	no
Reasoning Horizon (baseline)	0.059	0.90	11	no
Order fidelity (ours)	0.261	0.97	0	yes

Table 3: How our approach compares with every prior faithfulness test discussed in Section 2.

Paper	Reference used	Extra model calls	Intervenes on computation	Testable on empty CoT
Turpin et al. (2023)	CoT vs. itself (hint)	no	no	no
Lanham et al. (2023)	CoT vs. edited CoT	yes	no	no
Bentham et al. (2024)	CoT vs. edited CoT	yes	no	no
Tanneru et al. (2024)	CoT vs. edited CoT	yes	no	no
Ye et al. (2026)	CoT vs. edited CoT	yes	no	no
Zhong et al. (2026)	graph inferred from hidden states	no	no	no
Zhang et al. (2026)	graph written by a language model	no	no	no
Our work	exact solver graph $G^\star$	no	yes (DTMI)	yes

on the changed one. The final answer changes in 71%, 76%, and 82% of the cases for Qwen3-8B, Llama-3.1-8B, and Yi-1.5-9B. Models can react to an error inside the computation, but their reaction does not stay inside the dependencies that the solver defines.

4.5 Comparison with Baseline Metrics

We first place our approach next to every faithfulness test discussed in Section 2 on the properties that separate them (Table 3). Every earlier test either compares a CoT only with an edited copy of itself, or with a graph guessed or written by a person or another language model; none intervenes on the computation itself, and none can be checked on a CoT with no computation at all. The baseline formulas (Equation 1) have the model's own answers on both sides of the comparison; ours (Equations 2 and 3) have the solver's graph, which is what lets them say what *should* have happened.

The two kinds of metrics do not rank the tiers the same way: the baselines call Tier 3 least faithful, while ours find the dynamic-programming tier written down best and Tiers 1 and 3 equal – expected, since the metrics ask different questions, and exactly where ours add information. We see four reasons to prefer them when the computation is known. *A reference shows the difference between reacting and confining:* an answer-flip test calls a model faithful whenever its answer changes after a mistake is inserted; under DTMI the answer changes in 71–82% of cases, so a flip-based test would call all three models faithful, yet our PF is only 0.21–0.26, because the reaction also reaches 65–74% of the values that should not move; only a reference graph tells us which values *should* change. *Our metrics follow the computation:* adding the graph numbers raises order fidelity's R^2 from 0.15 to 0.41 (to 0.37 on unseen models), while the same step raises the flip-based metrics only from 0.041 to 0.083 (early answering) and 0.102 to 0.114 (Reasoning Horizon), reaching just 0.03 and 0.02 on

unseen models. *They add information and do not repeat it:* the two baseline tests agree with each other (Spearman correlation $+0.85$ and $+0.94$ on the two scored models; Appendix D) but are almost unrelated to order fidelity and PMF (correlation at most 0.28); order and propagation are properties the flip-based metrics do not observe. *A reference makes the metric testable:* because $G^\star$ is known, we could test our order metric on a CoT that computes nothing and correct it (Section 4.3); a metric that only compares a CoT with an edited copy of itself has no such reference. This also removes a known problem of flip-based tests: a model that reaches the right answer without using its CoT looks unfaithful by definition (Tanneru et al., 2024), while a graph-based score depends on the task, and answers the worry that metrics without true labels perform close to chance (Gur-Arieh et al., 2026).

What we found on their metrics: we ran the formulas above on exactly the same records as our metrics (Table 2, bottom), using the four families where order fidelity needs no correction (arithmetic, linear systems, LIS, edit distance) so every metric is scored on the same instances. Adding graph shape raises R^2 by 0.261 for order fidelity, versus only 0.053, 0.014, and 0.059 for early answering, mistake-ignored, and the Reasoning Horizon. The gap is larger on unseen models: order fidelity reaches a held-out R^2 of 0.378 against at most 0.064 for the baselines, and its per-cell means agree best between seeds ($r = 0.97$ against 0.67–0.90). Our metrics therefore gave the stronger, more informative results, while the baselines remain useful for what they were built to measure: how much the answer depends on the written steps.

5 Discussion and Future Directions

A complexity class describes the worst case of a whole family of problems, while faithfulness is a behaviour of one instance, and we find it follows the graph of that instance. Propagation Fidelity gives a second view: a faithful model should react strongly to an error without leaking it, but the models we tested react strongly *and* leak a lot, so reacting and confining are different behaviours. The null-CoT test teaches a practical lesson: any metric that matches text to graph nodes should be reported alongside a no-computation baseline and an order score that needs no global ranking.

A hand check of CYK transcripts, an order metric matching on node position, a second P-complete family, and instances that vary width, depth, or treewidth directly at a fixed size would sharpen the hardest-family results and test the instance-level mechanism of Section 3.4; a version of DTMI that breaks the model's own CoT, instead of a written list, would confirm the leakage comes from its own reasoning. The methods from earlier work could be extended to larger test sets, API-only and larger models, and models with hidden thinking traces; science and logic problems with checkable graphs, plus finer graph features, may give sharper predictions, and the intervention idea could be used for training by penalising errors that reach or miss the wrong values.

6 Conclusion

We introduced a computation-graph framework for examining whether, and where, a language model's chain-of-thought respects the exact computation graph of the problem it solves. By formalizing five families of computation as solver-built graphs $G^\star$ and measuring order fidelity, path-monotonic fidelity, and Propagation Fidelity against them, we showed that graph shape explains faithfulness beyond the difficulty tier, including for unseen models and after controlling for answer correctness and CoT length; an effect that comes mainly from how graph shape changes with problem size and task family. Deliberate corruption showed models react to an error without confining it to its dependents, and a CoT with no computation showed step-order metrics need a sanity test of their own. These findings position computation graphs as a useful reference for judging CoT faithfulness, shifting attention from comparing a CoT with itself to comparing it with the structure of the computation it claims to perform.

Reproducibility Statement

We release the exact $G^\star$ generators for all five families, the sweep and DTMI code, every per-instance record ($4{,}283$ records across 7 models and 2 seeds), and the analysis code (nested F-tests, permutation tests, leave-one-model-out, controls, bootstrap, figures). Every datapoint is regenerable from `(family, seed, scale, index)`.

REFERENCES

01.AI. Yi: Open foundation models by 01.ai. *arXiv preprint arXiv:2403.04652*, 2024.

Alireza Amiri, Xinting Huang, Mark Rofin, and Michael Hahn. Lower bounds for chain-of-thought reasoning in hard-attention transformers. In *International Conference on Machine Learning (ICML)*, 2025.

Iván Arcuschin, Jett Janiak, Robert Krzyzanowski, Senthooran Rajamanoharan, Neel Nanda, and Arthur Conmy. Chain-of-thought reasoning in the wild is not always faithful. *arXiv preprint arXiv:2503.08679*, 2025.

Oliver Bentham, Nathan Stringham, and Ana Marasović. Chain-of-thought unfaithfulness as disguised accuracy. *Transactions on Machine Learning Research*, 2024.

Paul C. Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. Thought anchors: Which llm reasoning steps matter? *arXiv preprint arXiv:2506.19143*, 2026.

Jonah Brown-Cohen, David Lindner, and Rohin Shah. Quantifying the necessity of chain of thought through opaque serial depth. *arXiv preprint arXiv:2603.09786*, 2026.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-snli: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, et al. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Lizhou Fan, Wenyue Hua, Lingyao Li, Haoyang Ling, and Yongfeng Zhang. NPHardEval: Dynamic benchmark on reasoning ability of large language models via complexity classes. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4092–4114, 2024.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Yoav Gur-Arieh, Ana Marasović, and Mor Geva. Faithfulness metrics don't measure faithfulness: A meta-evaluation with ground truth. *arXiv preprint arXiv:2605.25052*, 2026.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations (ICLR)*, 2024.

Albert Q. Jiang et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Maurice G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

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

Tomek Korbak, Mikita Balesni, Elizabeth Barnes, Yoshua Bengio, et al. Chain of thought monitorability: A new and fragile opportunity for AI safety. *arXiv preprint arXiv:2507.11473*, 2025.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. *International Conference on Learning Representations (ICLR)*, 2024.

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

Llama Team, AI @ Meta. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Uzay Macar, Paul C. Bogdan, Senthooran Rajamanoharan, and Neel Nanda. Thought branches: Interpreting llm reasoning requires resampling. *arXiv preprint arXiv:2510.27484*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. In *Transactions of the Association for Computational Linguistics (TACL)*, 2023.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *International Conference on Learning Representations (ICLR)*, 2024.

Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2009.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

Qwen Team. Qwen3 technical report. *arXiv preprint*, 2026.

Ansh Radhakrishnan, Karina Nguyen, Anna Chen, Carol Chen, Carson Denison, Danny Hernandez, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, et al. Question decomposition improves the faithfulness of model-generated reasoning. *arXiv preprint arXiv:2307.11768*, 2023.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2019.

Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 3rd edition, 2012.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics (ACL)*, 2023.

Sree Harsha Tanneru, Dan Ley, Chirag Agarwal, and Himabindu Lakkaraju. On the difficulty of faithful chain-of-thought reasoning in large language models. In *ICML 2024 Workshops (TiFA)*, 2024. arXiv:2406.10625.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2): 189–201, 1979.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Junda Wu, Xintong Li, Ruoyu Wang, Yu Xia, Yuxin Xiong, Jianing Wang, Tong Yu, Xiang Chen, Branislav Kveton, Lina Yao, Jingbo Shang, and Julian McAuley. OCEAN: Offline chain-of-thought evaluation and alignment in large language models. In *International Conference on Learning Representations (ICLR)*, 2025.

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

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

Donald Ye, Max Loffgren, Om Kotadia, Linus Wong, and Jonas Rohweder. Mechanistic evidence for faithfulness decay in chain-of-thought reasoning. *arXiv preprint arXiv:2602.11201*, 2026. ICLR 2026 workshop.

Jiaqiao Zhang, Zhoujun Li, Raoyuan Zhao, Jian Lan, Thomas Seidl, Michael A. Hedderich, Hinrich Schütze, and Yihong Liu. Beyond input understanding: Diagnosing multilingual mathematical reasoning with directed acyclic trace graphs. *arXiv preprint arXiv:2605.27715*, 2026.

Yizhou Zhang, Lun Du, Defu Cao, Qiang Fu, and Yan Liu. An examination on the effectiveness of divide-and-conquer prompting in large language models. *arXiv preprint arXiv:2402.05359*, 2024.

Tianjun Zhong, Linyang He, Ziyang Li, and Nima Mesgarani. From chains to DAGs: Probing the graph structure of reasoning in LLMs. *arXiv preprint arXiv:2601.17593*, 2026.

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

A Task Family Definitions

Table 4 gives, for each of the five task families, what a node in the exact graph $G^\star$ represents and when two nodes are connected by an edge.

Table 4: The five task families and how their exact graph $G^\star$ is built: what a node is, and when two nodes are connected.

Family	Complexity regime	Nodes in $G^\star$	Edges / dependencies in $G^\star$
Modular Arithmetic (prime p)	Bounded-depth (TC0/NC1)	Expression nodes: literals and values from binary ops	Operation node links to its operand nodes, forming an expression-evaluation tree
Modular Linear Systems	Bounded-depth (TC0/NC1)	Pivot/elimination results during Gaussian elimination	Edge to the earlier pivot/elimination result it depends on
Longest Increasing Subsequence (LIS)	General DP	One node $dp[i]$ per position i	$j \to i$ iff $j < i$, $a[j] < a[i]$, $dp[j]{+}1 \leq dp[i]$; critical path = an LIS
Edit Distance	General DP	Lattice node (i, j): edit-distance state for prefixes	Edges from $(i{-}1, j)$, $(i, j{-}1)$, $(i{-}1, j{-}1)$: deletion, insertion, substitution
CFG Membership (CYK)	P-complete	Chart cell $(span, i)$: substring/span computation	Edges from $(split{-}1, i)$ and $(span{-}split, i{+}split)$ per split

B Summary of Results

This paper runs several experiments, in the main text and the appendix. Table 5 gathers the main number from every one of them into groups: faithfulness by complexity tier; how much graph geometry adds on top of the tier; behaviour under DTMI; the head-to-head comparison against

baseline metrics; results under alternative reasoning strategies; interventions from earlier work; and agreement with Lanham's own scores. Table 6 gives the four robustness checks of Section 4.2.

Table 5: All results from every experiment in this paper, gathered into one table.

Faithfulness by complexity tier, pooled over all models and seeds					
Tier	n	Accuracy	Early-ans. unf.	Mistake-ign.	OF
1 (bounded-depth)	1877	0.46	0.17	0.07	+0.19
2 (dynamic prog.)	1680	0.18	0.23	0.19	+0.52
3 (P-complete, CYK)	726	0.34	0.35	0.34	+0.17
Effect of adding graph geometry (depth, width, treewidth) on top of tier + (model, seed)					
Measure	R^2: tier+FE $\rightarrow$ +geometry		F	Held-out R^2 (4-family subset)	
Early-answer unfaith.	$0.041 \rightarrow 0.083$		65.3	$-0.007 \rightarrow 0.024$	
Order fidelity	$0.148 \rightarrow 0.409$		560.9	$0.126 \rightarrow 0.378$	
Reasoning Horizon	$0.102 \rightarrow 0.114$		19.2	$0.017 \rightarrow 0.064$	
Behavior under DTMI: corrupting one intermediate value and watching the reaction					
Model	n (nodes)	PF	Sensitivity	Leakage	Answer changed
Qwen3-8B	360	0.26	0.79	0.65	71%
Llama-3.1-8B	360	0.21	0.82	0.74	76%
Yi-1.5-9B	185	0.22	0.80	0.72	82%
Head-to-head: our order fidelity vs. the baseline metrics, four-family subset, same records					
Metric	Gain in R^2	Seed agreement (r)		Extra calls	Empty-CoT test
Early-ans. (baseline)	0.053	0.90		3	no
Mistake-ign. (baseline)	0.014	0.67		1	no
Reas. Horizon (baseline)	0.059	0.90		11	no
Order fidelity (ours)	0.261	0.97		0	yes
Alternative reasoning strategies: OF and accuracy by tier (Table 7, Appendix C)					
Strategy	Tier 1 OF	Tier 1 acc.	Tier 2 OF	Tier 2 acc.	Tier 3 OF / acc.
Plain CoT	0.19	0.46	0.52	0.18	+0.17 / 0.34
Tree-of-Thought	0.132	0.357	0.552	0.238	+0.49 / 0.524
Divide-and-Conquer	0.278	0.073	0.401	0.042	+0.06 / 0.042
Length-CoT (s/m/l)	0.213 / 0.249 / 0.234		0.531 / 0.528 / 0.530		0.15 / 0.16 / 0.05
Interventions from earlier work on Llama-3.1-8B, CYK (Table 9, Appendix E)					
Condition	Order fidelity		PMF	Accuracy	Tests
Zero-shot	+0.28		0.72	0.00	6
ICL, random demos	+0.79		0.93	0.00	6
ICL, highest-OF demos	+0.79		0.97	0.00	6
LoRA (best selection)	+0.80		1.00	0.00	3
Agreement between Lanham's own tests and our scores, Spearman (Table 8, Appendix D)					
Comparison	Qwen3-8B			Llama-3.1-8B	
EA AOC vs. corrected OF	+0.06			−0.09	
AM AOC vs. corrected OF	+0.28			−0.23	
EA AOC vs. AM AOC	+0.85			+0.94	

Table 6: Robustness of the geometry effect: R^2 before $\rightarrow$ after adding geometry, by control, with F of the nested test. For fixed size, the shuffle test gives $p = .70$, .70, and .02.

Control	Order fidelity	Early-answer unfaith.	Reasoning Horizon
+ Correctness	$0.153 \rightarrow 0.412$ (F=561)	$0.046 \rightarrow 0.089$ (F=67)	$0.546 \rightarrow 0.551$ (F=16)
+ CoT length	$0.172 \rightarrow 0.410$ (F=513)	$0.042 \rightarrow 0.090$ (F=74)	$0.133 \rightarrow 0.141$ (F=15)
Fixed size	$0.548 \rightarrow 0.549$ (F=2.3)	$0.141 \rightarrow 0.141$ (F=0.3)	$0.211 \rightarrow 0.212$ (F=2.7)
+ Family	$0.326 \rightarrow 0.519$ (F=381)	$0.097 \rightarrow 0.133$ (F=45)	$0.115 \rightarrow 0.187$ (F=95)

C Alternative Reasoning Strategies

Does the way reasoning is asked for change how well the CoT follows the graph? We repeated the order measurement with three other strategies. Tree-of-Thought (ToT) (Yao et al., 2023) lets the model search over next steps. Divide-and-Conquer (DaC) (Zhang et al., 2024) splits the problem into sub-tasks. Length-conditioned CoT (Wu et al., 2026) asks for a short, medium, or long chain ($0.5\times$, $1\times$, or $2\times$ the usual length). All results are pooled over the seven models. DaC and length-conditioned CoT use both seeds, and ToT uses seed 2024 only, decided from measured time per instance under a fixed compute budget.

Table 7: Order fidelity and accuracy by tier, for plain CoT and three alternative reasoning strategies: Tree-of-Thought (Yao et al., 2023), Divide-and-Conquer (Zhang et al., 2024), and Length-CoT (Wu et al., 2026).

Strategy	Tier 1 OF	Tier 1 acc.	Tier 2 OF	Tier 2 acc.	Tier 3 OF (masked)	Tier 3 acc.
Plain CoT (Table 1)	0.19	0.46	0.52	0.18	+0.17	0.34
Tree-of-Thought	0.132	0.357	0.552	0.238	$+0.49^{\dagger}$	0.524
Divide-and-Conquer	0.278	0.073	0.401	0.042	+0.06	0.042
Length-CoT (short)	0.213	0.428	0.531	0.211	+0.15	0.300
Length-CoT (mid)	0.249	0.420	0.528	0.161	+0.16	0.253
Length-CoT (long)	0.234	0.420	0.530	0.195	+0.05	0.184

For Tier 3, order fidelity is $+0.17$ for plain CoT and between $+0.05$ and $+0.16$ for DaC and the three length settings ($n = 30$–49 transcripts each). No strategy reverses the order and none clearly improves it. DaC lowers accuracy on every tier (0.07, 0.04, 0.04 against 0.46, 0.18, 0.34 for plain CoT). This matches its own theory: Zhang et al. (2024) say it helps only when the sub-tasks are independent, and every family here is sequential. A longer CoT does not reliably help, and it lowers Tier-3 accuracy ($0.300 \to 0.253 \to 0.184$ from short to long), as Wu et al. (2026) describe for long chains. ToT is the only strategy that raises Tier-3 accuracy (0.524 against 0.34).

D Agreement with Lanham's Scores

We ran the full versions of the two tests of Lanham et al. (2023) on the stored CoTs of Qwen3-8B and Llama-3.1-8B. Early-answering AOC cuts the chain after every step (up to 20 cut points). Adding-mistakes AOC changes a number (or a nonterminal id for CYK) in step k and asks for the answer immediately. Table 8 shows how they relate to our scores. We use the Spearman correlation, which compares rankings.

Table 8: Agreement between Lanham's own tests and our scores, for two models.

Spearman correlation	Qwen3-8B	Llama-3.1-8B
early-answering AOC vs. corrected order fidelity	$+0.06$ (n=40)	-0.09 (n=36)
early-answering AOC vs. edge-wise PMF	-0.01	-0.07
adding-mistakes AOC vs. corrected order fidelity	$+0.28$ (n=34)	-0.23 (n=26)
early-answering AOC vs. our 3-point early-answering rate	-0.57	-0.67
early-answering AOC vs. adding-mistakes AOC	$+0.85$ (n=7)	$+0.94$ (n=6)

E Interventions from Earlier Work

Tanneru et al. (2024) try to raise faithfulness with in-context learning (ICL, examples placed in the prompt), fine-tuning, and activation editing. We ran a small version of each on Llama-3.1-8B, with CYK test instances, and scored the outputs with our metrics. Demonstrations and training examples come from seed 2024 and are not filtered for correctness, because the model solved none of its CYK instances. Demonstrations are picked at random (DU), by the lowest early-answering rate as in their

most-faithful choice (DF), or by the highest corrected order fidelity (DO). Fine-tuning uses low-rank adaptation (LoRA), which trains a small add-on instead of the whole model (rank 8 on the query and value projections, 3 epochs, 10 examples), on DF or DO examples. Activation editing shifts the 48 attention heads that linear probes link to faithfulness, at two strengths.

Table 9: Interventions of prior work on Llama-3.1-8B, CYK, scored with our metrics.

Condition	Tests	Order fidelity	PMF	Accuracy
zero-shot	6	$+0.28$	0.72	0.00
in-context, random demos (DU)	6	$+0.79$	0.93	0.00
in-context, lowest early-answering rate (DF)	6	$+0.66$	0.84	0.00
in-context, highest order fidelity (DO)	6	$+0.79$	0.97	0.00
LoRA, DF selection	3	$+0.73$	0.87	0.00
LoRA, DO selection	3	$+0.80$	1.00	0.00
activation editing, two strengths	3 each	outputs degenerate into repetition; no order score		

Demonstrations and fine-tuning raise order fidelity from $+0.28$ to about $+0.7$–$+0.8$ and PMF from 0.72 to 0.84–1.00, and accuracy stays the same. The choice by faithfulness is not better than a random choice. The probes reach an accuracy of 0.94–0.97 on held-out data. The steered outputs start to repeat themselves (for example "ac, ac, ac"), so no order score exists for them.

F Statistical Details

Table 10 gives the per-model order fidelity and PMF scores on the hardest (CYK) family referenced in Section 4.3.

Table 10: Order fidelity and PMF on the hardest (CYK) family, per model, before and after hiding the repeated input string.

Model	OF, restated-input matcher	OF, input masked [95% CI]	PMF	n
Llama-3.1-8B	$+0.04$	$+0.41$ $[+0.32, +0.50]$	0.76	112
Mistral-7B-v0.3	$+0.07$	$+0.35$ $[+0.24, +0.44]$	0.71	63
Qwen2.5-7B	-0.26	$+0.23$ $[+0.07, +0.37]$	0.64	52
Qwen3-8B	-0.36	$+0.12$ $[-0.02, +0.26]$	0.60	87
Qwen3-4B	-0.37	$+0.05$ $[-0.11, +0.22]$	0.61	60
Yi-1.5-9B-Chat	-0.38	-0.01 $[-0.14, +0.12]$	0.54	82
Qwen3-32B	-0.45	-0.33 $[-0.54, -0.11]$	0.45	29

Nested F-test. The base model predicts a score from an intercept, tier indicators, and (model, seed) indicators. The full model adds depth, width, and treewidth. The F value is $F = \frac{(\mathrm{RSS}_0 - \mathrm{RSS}_1)/3}{\mathrm{RSS}_1/(n-p_1)}$, where RSS is the sum of squared errors and p_1 is the number of parameters of the full model.

Shuffle (permutation) test. We shuffle the graph numbers among instances of the same (model, seed, tier) group (200–300 times) and compute $p = (\#\{F^* \geq F\} + 1)/(N + 1)$, where F^* comes from a shuffle and N is the number of shuffles.

Unseen-model test (leave-one-model-out). For each model we fit on the other six and evaluate on the held-out model with $R^2 = 1 - \mathrm{SSE}/\sum(y - \bar{y}_{\mathrm{train}})^2$. The family baseline uses family indicators instead of tier indicators.

Checks. Check 1 adds an indicator for a correct answer. Check 2 adds the logarithm of the CoT length in characters. Check 3 replaces the tier indicators by family$\times$scale indicators. Check 4 adds family indicators to a model with depth$\times$width quartile cells and treewidth.