# CEILING OF A TASK: WHEN CAN A TRANSFORMER SUCCEED WITHOUT ITS CHAIN OF THOUGHT?

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Reasoning models generate long chains of thought before they answer, yet it is debated whether the content of these chains does real computational work or is largely decorative. We study this question by viewing a transformer as a shallow circuit. One forward pass through a fixed number of layers has constant depth, so any procedure that runs the model a constant number of times is a shallow circuit. We call the best accuracy that a shallow circuit can reach on a task the *ceiling* of the task, and a task is serial if its ceiling lies below one. We prove three results on serial tasks that hold for every transformer, no matter how it was trained. *Necessity*: replacing the chain by anything that does not depend on its content, such as filler tokens or a restatement of the question, drives the accuracy down to the ceiling, and on a maximally serial task down to chance. *Depth*: no shallow computation can write the chain of a model whose accuracy exceeds the ceiling, not even approximately. *Locality*: the answer is one shallow pass away from the finished chain, so all of the serial reasoning happens in the chain. On word problems of finite groups, whose ceilings are known, small transformers trained from scratch, with or without reinforcement learning, attain the predicted numbers: chain-trained models solve every input length and fall to chance when the chain is erased, chainless models collapse to the ceiling as the input length grows, and open-weight reasoning models given the same problem in words return to the baseline without their chain. On MATH-500 and AIME, erasing the chain costs open reasoning models 0.52 to 0.82 accuracy, a sentence shuffle is harmless, and a token shuffle is as harmful as erasing; the same holds for checkpoints trained by GRPO with a correct or a random reward. The ceiling of a task therefore answers when a transformer can succeed without its chain of thought, no matter how the model was trained.

## 1 INTRODUCTION

Reasoning models produce long chains of thought, and longer chains are often correlated with better performance on hard problems. But it is unclear whether these chains do real computational work or are mostly decorative, as recent work shows that filler and pause tokens can reproduce some gains (Pfau et al., 2024; Goyal et al., 2024), and chains often fail to state the true reasons for answers (Turpin et al., 2023; Chen et al., 2025). Besides, experiments that erase, truncate, or shuffle chains (Lanham et al., 2023) are hard to interpret because it is unclear whether the accuracy drops are due to the chain or the model (answer head) reading it. This paper proves quantitative bounds on what any transformer can do without the content of its chain, no matter how it is trained.

To study how CoT improves information processing in LLMs when handling complex tasks, we model one forward pass of a transformer with $L$ layers as a calculation conducted by a circuit of constant depth $L$, where "constant" means that the amount of parallelizable computation does not depend on the input size, and we call such a circuit a *shallow circuit*. As shown in Figure 1 (left), increasing the input length and context window expands the circuit's width, but never increases the information processing depth. Depth can only be increased by producing a token and then feeding it back in as part of the input. As a result, generating $T$ output tokens causes the input information to be processed through $T$ sequential forward passes, rather than a single pass when there is no CoT. The intermediate tokens are the links that carry information forward from one pass to the next across the $T$ passes.



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.

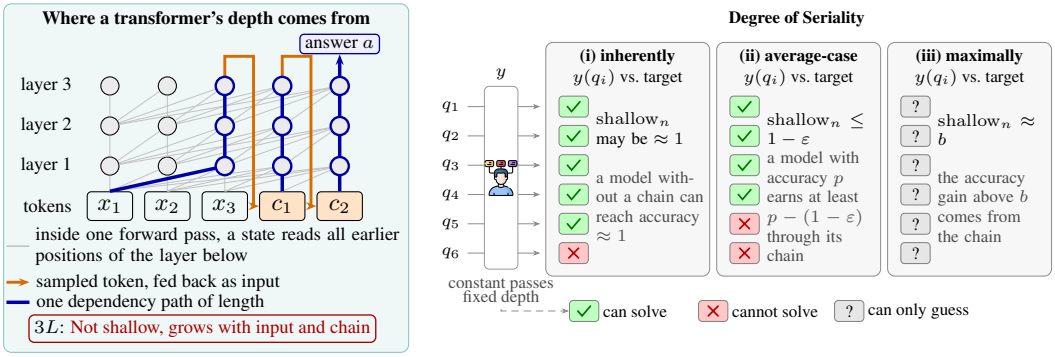


Figure 1: *Left:* a transformer with $L = 3$ layers on three input tokens and two generated ones. Inside one forward pass a hidden state reads only earlier positions of the layer below, so information travels at most $L$ steps. A transformer is therefore *shallow* when its output length is constant and not when it grows with the input. *Right:* the three degrees of seriality of Definition 2.4 for a task: a collection of inputs $q_i$ with targets $y(q_i)$.

This is what makes reasoning powerful: with polynomially many intermediate tokens, a transformer can solve problems that no single forward pass can (Li et al., 2024; Merrill & Sabharwal, 2024). Existing results of this type, based on the circuit perspective on transformers (Merrill & Sabharwal, 2023; 2025; Amiri et al., 2025), establish that for a given class of problems there exists a transformer with some chain length that solves it. We ask a different question, about one transformer that has already been trained by any algorithm: *what can it do at test time without the content of its chain?*

A fixed transformer run a constant number of times is itself a shallow circuit (Lemma 2.3), so its accuracy is bounded by the best accuracy any shallow circuit reaches on the task, the task's *ceiling* $\text{shallow}_n$. A task is defined as *average-case serial* if its ceiling is bounded away from one and *maximally serial* if its ceiling is the guessing baseline (Section 2); for the word problem of the group $A_5$, the ceiling is the guessing baseline by a result of Miles & Viola (2013), which makes it the test case of Section 6. Applying the lemma at the three places of Figure 2 (right) gives three theorems that hold for every transformer, however it was trained. *Necessity* (Section 3): replace the chain by anything that does not depend on its content, such as filler tokens or a restatement of the question, and the accuracy falls to the ceiling, or to chance on a maximally serial task. *Depth* (Section 4): a chain that lifts accuracy above the ceiling cannot be written by any shallow computation, not even approximately, so its tokens must build on one another over a number of steps that grows with the input. *Locality* (Section 5): the answer head is one forward pass, so once the chain ends the answer is readable from it by a shallow circuit; all the serial work sits in the chain. None of this requires the chain to be faithful or human-readable. Appendix A discusses what the theorems say about current reasoning models, from one-pass models to process reward models.

We test the theory at two scales (Section 6). On word problems of finite groups, whose ceilings are known, small transformers trained from scratch, with or without reinforcement learning, match the predicted numbers: with a chain they reach accuracy 1 at every input length, and fall to chance when the chain is erased. Without a chain they collapse to the ceiling once the input outgrows their depth. Open-weight reasoning models given the same word problem in words also return to the baseline without their chain, and a chain of bounded length helps them less as the input grows. On AIME and MATH-500, where the ceiling is unknown, open reasoning models with 1.5B to 32B parameters lose 0.52 to 0.82 accuracy when the chain is erased. With only a prefix of the chain, the remaining accuracy depends on how often the kept prefix already states the answer. Their accuracy survives a shuffle of the sentences of the chain but not of its tokens, since a shallow decoder can still read the answer from the bag of sentences, and the same pattern holds for checkpoints trained by GRPO with a correct or a random reward. Together, they reveal that without its chain a transformer keeps exactly what one pass can compute, and everything above the ceiling of the task comes from the chain, whether the model was trained by supervision or by reinforcement learning.

Our contributions are summarized as follows.

- A framework that measures a task by its ceiling, the best accuracy of a shallow circuit. The ceiling bounds the accuracy that reasoning LLMs, small transformers, one-pass transform-

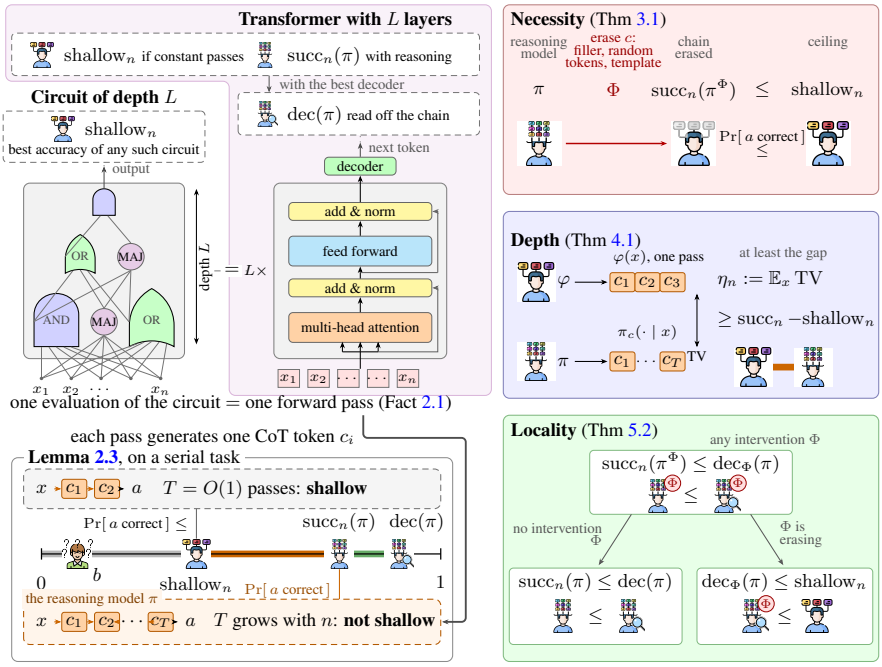


Figure 2: *Top left:* one forward pass of a transformer with $L$ layers can be viewed as a circuit of constant depth $L$ with unbounded fan-in (Fact 2.1). The ceiling $\text{shallow}_n$ ( ) is the best accuracy of any such shallow circuit, $\text{succ}_n(\pi)$ ( ) is the accuracy of the reasoning model, and $\text{dec}(\pi)$ ( ) is what the best shallow decoder reads off the finished chain. *Bottom left (Lemma 2.3):* a constant number of passes is still shallow and lands below the ceiling; only a chain that grows with the input can rise above it. *Right:* for every transformer, erasing the chain sends the accuracy back to the ceiling (Necessity), no shallow computation could write a chain close to the model's reasoning chain (Depth), and the finished chain already makes the answer readable in one pass (Locality).

ers and latent reasoners alike can reach without a chain. It also grades tasks by how much of the accuracy on them has to come from serial computation, which we call degrees of seriality (Section 2). We extend the worst-case notion of Li et al. (2024) to the average case.

- Three theorems on chains of thought, Necessity, Depth, and Locality, that hold for every transformer at test time regardless of how it was trained and bound what a trained transformer can do without the content of its chain (Sections 3 to 5).
- Experiments on tasks with a known ceiling and on math reasoning benchmarks: small transformers trained from scratch, with or without RL, and open-weight reasoning models land on the predicted ceiling without their chains, and interventions on open reasoning models on mathematical benchmarks, including erasing, truncating, and shuffling the chain, all behave exactly as the theorems predict (Section 6).

## 2 Transformers as Shallow Circuits

**Circuits and depth.** We use $n$ to denote the length of the input of a circuit, and $\text{poly}(n)$ is the set of all quantities bounded by a polynomial in $n$. The *depth* of a Boolean circuit is the length of the longest path from an input to the output. The class $\mathsf{TC}^0$ consists of circuit families of size $\text{poly}(n)$ that have constant depth and unbounded fan-in. The allowed gate types include AND, OR, NOT, and majority, and a majority gate enables counting within a single circuit layer. We call a circuit family *shallow* if it lies in $\mathsf{TC}^0$, or in $\mathsf{AC}^0$ (under the constant-precision setting, detailed in Appendix B.1). The forward pass of a transformer follows precisely this structure (Figure 2, top left), where its depth is fixed and does not depend on the input length $n$ (Figure 1, left).

**Fact 2.1** (One forward pass of a transformer is shallow; Merrill & Sabharwal, 2023)**.** *Take a transformer with a fixed (constant) depth, context length* $\text{poly}(n)$*, and* $O(\log n)$*-bit numerical precision. For any given input, the result of a single forward pass can be computed by a circuit family in* $\mathsf{TC}^0$*.*

In the following discussion, a transformer means a model as in Fact 2.1, and the theorems of Sections 3 to 5 are stated for such models.

**Pipelines and the ceiling.**

**Definition 2.2** (Pipeline)**.** Fix a transformer $\pi$. A *constant-pass pipeline* is a randomized procedure that maps an input $x$ to an answer in a *constant* number of stages. Each stage processes $x$ and the outputs of the earlier stages, either by applying a shallow randomized map or by running one forward pass of $\pi$. The output of the last stage is the answer.

Lemma 2.3 below shows that every pipeline is a shallow circuit (Figure 2, bottom left), so the best accuracy that a shallow circuit of the same depth and size reaches on a task bounds every pipeline on that task. For constants $d$ and $s$, let $\mathcal{C}_n(d, s)$ be the set of randomized circuits of depth at most $d$ and size at most $n^s$ built from the gates of $\mathsf{TC}^0$; a shallow randomized circuit family is a sequence $\{C_n\}$ with $C_n \in \mathcal{C}_n(d, s)$ for some fixed $d$ and $s$. For a task with target $y$ and input distribution $\mathcal{D}_n$, the *ceiling* of the task at depth $d$ and size $n^s$ is

$$\mathrm{shallow}_n(d, s) \;:=\; \sup_{C \in \mathcal{C}_n(d,s)} \Pr_{x \sim \mathcal{D}_n,\, r} \big[\, C(x, r) = y(x) \,\big],$$

where $r$ is the random string of the circuit. We write $\mathrm{shallow}_n$ without the two constants when they are clear from the context, and Appendix B states how such a bound is read.

**Lemma 2.3** (Pipelines are shallow)**.** *For every transformer $\pi$ and every constant-pass pipeline, there exist constants $d$ and $s$, depending only on $\pi$ and on the pipeline, and a family of shallow randomized circuits $C_n \in \mathcal{C}_n(d, s)$ whose output distribution on every input matches the pipeline's answer distribution. Consequently, on every task and for every input distribution, the success of the pipeline is at most* $\mathrm{shallow}_n(d, s)$.

We include the detailed proof in Appendix B.

**Three degrees of seriality.** It is therefore important to identify which tasks have an upper bound that is strictly below one, indicating that the tasks cannot be solved with only a constant number of forward passes. Based on the fraction of inputs that pipelines can solve, we define three levels of *seriality* as a property of the task (Figure 1, right).

**Definition 2.4** (Degrees of seriality)**.** For each input length $n$, a task consists of a target function $y$ and a distribution $\mathcal{D}_n$ over inputs, and the accuracy of any procedure on the task is the probability that it outputs $y(x)$ for $x \sim \mathcal{D}_n$. Let $b$ be the *guessing baseline* of the task. The task is

(i) *inherently serial*, in the sense of Li et al. (2024), if no shallow circuit family computes $y$ on every input;

(ii) *average-case serial* if its ceiling is bounded away from one, that is, if there is a constant $\varepsilon > 0$ such that for every $d$ and $s$, $\mathrm{shallow}_n(d, s) \le 1 - \varepsilon$ for all sufficiently large $n$, so that every pipeline fails on at least an $\varepsilon$ fraction of the inputs;

(iii) *maximally serial* if its ceiling is the guessing baseline up to a negligible term, that is, if for every $d$ and $s$ and every constant $k$, $\mathrm{shallow}_n(d, s) \le b + n^{-k}$ for all sufficiently large $n$, so that every pipeline does no better than guessing.

In (ii) and (iii) the threshold on $n$ may depend on $d$ and $s$.

Condition (i) is a worst-case notion and does not depend on the input distribution. Conditions (ii) and (iii) are average-case notions, since the ceiling is an accuracy averaged over the input distribution of the task.

Our argument focuses on the second and third degrees, because on a task that is only inherently serial, a shallow circuit can succeed on all but a vanishing fraction of the inputs, so a model without a chain can reach accuracy close to one. Figure 1 (right) shows how much of the accuracy of a model has to come from its chain at each degree. The serial degrees are properties of the task, and we study what they imply for every model that tries to solve it.

Note that the degrees are relative to the precision setting, because shallow means $\mathsf{TC}^0$ at log precision and $\mathsf{AC}^0$ at constant precision. The main text uses the log-precision setting, following Merrill & Sabharwal (2023; 2024), and Appendix B.1 gives the constant-precision version of every result in the setting of Li et al. (2024).

## 3 NECESSITY: THE ACCURACY RESTS ON THE CONTENT OF THE CHAIN

We first examine what Lemma 2.3 tells us about the necessity of CoT, by analyzing the performance drops upon replacing the chain after it has been produced. We argue that when the replacement does not depend on the content of the chain, the replacement and the answer head form a two-stage (constant-pass) pipeline, and the accuracy of every reasoning model thus drops to the ceiling of the task, no matter how it was trained.

**Interventions and the load-bearing gap.** Define an *intervention* $\Phi$ on the CoT as a shallow randomized procedure that takes the input $x$ and the generated chain $c$ and returns a modified chain $\tilde{c} = \Phi(x, c)$ of length at most $\mathrm{poly}(n)$. The *intervened model* $\pi^{\Phi}$ runs $\pi$ as a reasoning model, for $T$ forward passes to generate $c$, then replaces $c$ by $\tilde{c}$ and feeds it into the prompt to sample the answer from $\pi(\cdot \mid x, \tilde{c})$, as illustrated by Figure 6 in Appendix D. We study the *load-bearing gap* $\Delta_{\Phi}(\pi) := \mathrm{succ}_n(\pi) - \mathrm{succ}_n(\pi^{\Phi})$, the accuracy that the model loses when its chain is replaced. An intervention is *erasing* if it does not read the chain, that is, $\Phi(x, c) = \varphi(x)$ for a shallow randomized map $\varphi$ (Figure 2, right). Interventions that read the chain, such as truncation and shuffling, are treated in Section 5.

**Theorem 3.1** (Necessity)**.** *Let $\pi$ be any transformer and let $\Phi$ be any erasing intervention. There are constants $d$ and $s$, depending only on $\pi$ and $\Phi$, such that on every task and for every input distribution,* $\mathrm{succ}_n(\pi^{\Phi}) \le \mathrm{shallow}_n(d, s)$*, and hence* $\Delta_{\Phi}(\pi) \ge \mathrm{succ}_n(\pi) - \mathrm{shallow}_n(d, s)$.

On a maximally serial task, the accuracy of a model rests entirely on the content of its chain. Training can only change what the model writes in its chain. It can never produce a model that keeps its accuracy without reasoning, because the theorem constrains what any transformer can compute at test time.

Appendix A discusses what our theorems imply about current reasoning models, including one-pass models such as Jev, latent reasoning, diffusion decoding, chain compression, and process reward models.

## 4 DEPTH: NO SHALLOW COMPUTATION CAN WRITE THE CHAIN

In Section 3, Theorem 3.1 bounds the accuracy of a model after its chain has been replaced. In this section, we ask what happens if a shallow map produces chains close to the model's own. We argue that the accuracy changes by at most the distance between the two chain distributions, and the swapped-in version is capped by the ceiling, so the chain of a successful reasoning model cannot be replaced by a shallow computation.

Define a *shallow imitator* of $\pi$ to be a shallow randomized family $\varphi$ that maps an input $x$ to a chain, and its *imitation error* $\eta_n := \mathbb{E}_x \ \mathrm{TV}\left(\varphi(x), \pi_c(\cdot \mid x)\right)$ is the average distance between its chains and the chains $\pi_c(\cdot \mid x)$ that the model generates. Here $\mathrm{TV}(P, Q) := \max_A |P(A) - Q(A)|$ is the total variation distance, the largest difference in probability that $P$ and $Q$ assign to any set of chains; it is 0 when the two distributions are identical and 1 when no chain can be produced by both.

**Theorem 4.1** (Depth)**.** *Let $\pi$ be any transformer and let $\varphi$ be any shallow imitator of $\pi$ with imitation error $\eta_n$. There are constants $d$ and $s$, depending only on $\pi$ and $\varphi$, such that on every task and for every input distribution,* $\mathrm{succ}_n(\pi) - \mathrm{shallow}_n(d, s) \le \eta_n$.

Because the total variation distance between a chain produced by shallow generation and the original chain from the successful reasoner is at least as large as the performance gap between that reasoner and a pipeline, it follows that the output chains of an accurate model are not even approximately close to the set of chains that any shallow computation can produce. Shallow computation can only generate chains where each token depends on previous tokens through a constant-bounded number of rounds. This further indicates that a successful model has to produce chains whose tokens are built on one another over a number of rounds that increases with the input.

## 5 LOCALITY: ALL OF THE MULTI-STEP WORK HAPPENS IN THE CHAIN

Finally, we apply Lemma 2.3 to the answer head. It produces the answer with one forward pass after the chain ends, so it is shallow and cannot contribute any serial work. We therefore conclude that

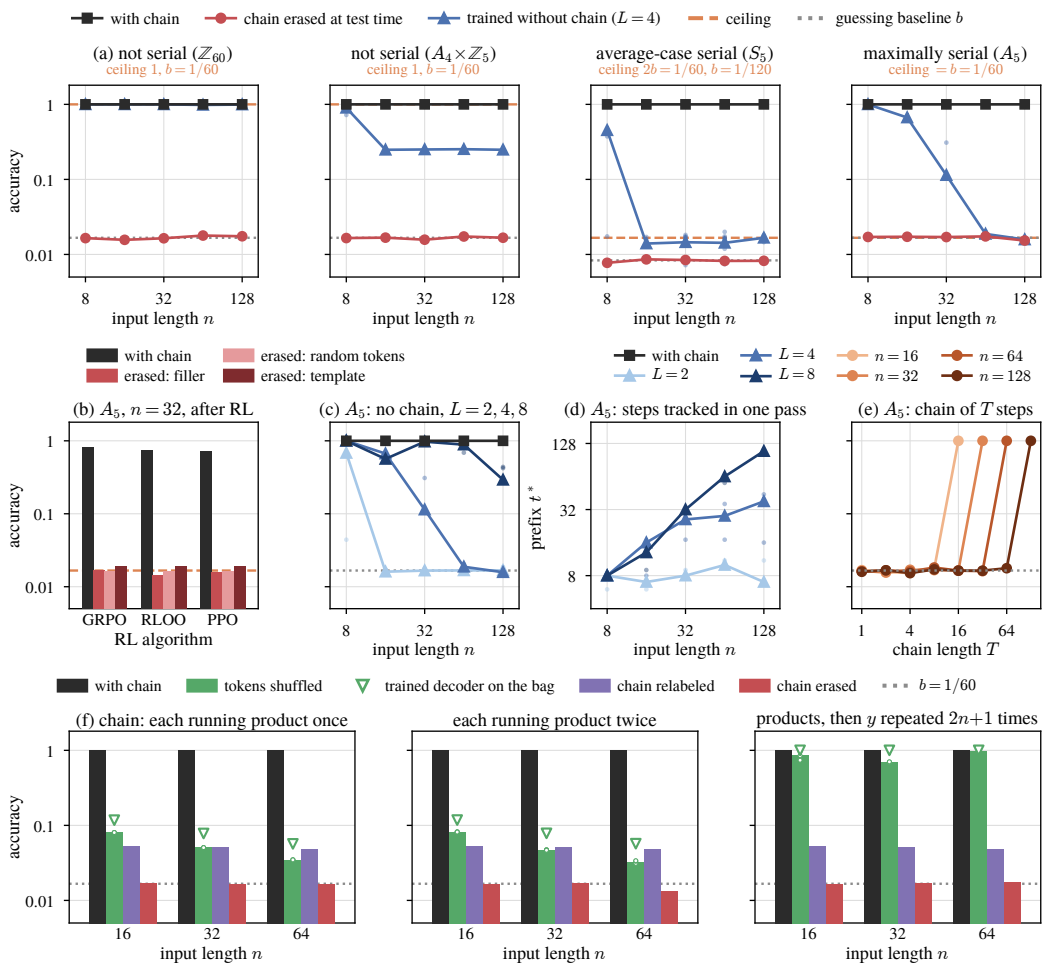


Figure 3: Small transformers on word problems ($L = 4$ unless stated; $n = 8$ to 128). *(a)* Necessity, from the least to the most serial group: models trained with the chain solve every length (black); erasing their chain at test time drops them to the baseline $b$ on every group (Theorem 3.1). *(b)* The same drop, for each of the three erasing interventions, after reinforcement learning with GRPO, RLOO, or PPO on $A_5$ at $n = 32$. *(c)* Depth: without a chain, accuracy stays high only while one pass can track the input, and the collapse length grows with $L$. *(d)* The tracked prefix $t^*$ levels off at a value set by $L$. *(e)* Models trained to write a chain of $T$ steps: only $T = n$ solves the task. *(f)* Locality on $A_5$ for three chain formats. After a token shuffle the answer head reads only the bag of chain tokens, so its accuracy (green) cannot exceed that of the best bag decoder trained on the bag ($\triangledown$) (Theorem 5.2, Lemma F.1). The bag reveals the answer only when the chain repeats it $2n + 1$ times (Proposition F.2).

by the time the chain is generated, the answer must already be computable by a shallow circuit from the input and the finished chain.

**Definition 5.1** (Shallow decodability)**.** A *decoder* is a shallow randomized circuit that predicts the target $y(x)$ from the input $x$ and a chain. For a transformer $\pi$, an intervention $\Phi$, and constants $d$ and $s$, the *shallow decodability* $\mathrm{dec}_\Phi(\pi; d, s)$ is the highest accuracy that any decoder in $\mathcal{C}_n(d, s)$ reaches on $x$ and the intervened chain $\Phi(x, c)$, where $c$ is the chain that the model generates; $\mathrm{dec}(\pi; d, s)$ denotes the same quantity when no intervention is applied, which is the highest accuracy that any such decoder reaches from $x$ and the chain the model generates. As for the ceiling, we write $\mathrm{dec}_\Phi(\pi)$ and $\mathrm{dec}(\pi)$ when the two constants are clear from the context.

**Theorem 5.2** (Locality)**.** *For every transformer $\pi$ there are constants $d$ and $s$, depending only on $\pi$, such that for every intervention $\Phi$, every task, and every input distribution,* $\mathrm{succ}_n(\pi^\Phi) \le \mathrm{dec}_\Phi(\pi; d, s)$*; in particular* $\mathrm{succ}_n(\pi) \le \mathrm{dec}(\pi; d, s)$*. Moreover, when the intervention $\Phi$ is erasing (Section 3), for every $d$ and $s$ there are constants $d'$ and $s'$, depending only on $d$, $s$, and $\Phi$, such that* $\mathrm{dec}_\Phi(\pi; d, s) \le \mathrm{shallow}_n(d', s')$.

The theorem tells us when a reasoning model has its answer. The answer is committed by the time the chain ends, in the sense that a shallow computation with no reasoning ability of its own could read it from the input and the finished chain.

## 6 EXPERIMENTS

We conduct two sets of experiments: (1) We train small transformers on word problems of finite groups, whose ceilings are known, so every intervened accuracy in Sections 3 through 5 has an exact predicted value. (2) We apply the interventions of Sections 3 and 5 to open-source reasoning models, first on the word problem of $A_5$, and then on mathematical benchmarks, whose ceiling is unknown. The full setup is given in Appendix G, and additional results are given in Appendix H.

### 6.1 TASKS WITH A KNOWN CEILING: SMALL TRANSFORMERS TRAINED FROM SCRATCH ON WORD PROBLEMS

For a finite group $G$, the word problem $\mathrm{WP}_G$ at length $n$ asks for the product $y(x) = g_1 g_2 \cdots g_n$ of an input $x = (g_1, \ldots, g_n)$ drawn uniformly from $G^n$ unless stated otherwise, with each $g_i$ encoded as one token, so $b = 1/|G|$, and a chain that writes the running product once per element solves it exactly. Its ceiling is known, and it reaches each of the three degrees when defined on different groups (Liu et al., 2023; Merrill et al., 2024), as detailed in Appendix C.2: the maximally serial case is $\mathrm{WP}_n$, the word problem of $A_5$ on uniform inputs; the average-case serial case is the word problem of $S_5$; and the non-serial cases, whose ceiling is 1, are the solvable groups $\mathbb{Z}_{60}$ and $A_4 \times \mathbb{Z}_5$ (Table 2 in Appendix D). In the maximally serial case the ceiling is $\frac{1}{60} + n^{-k}$ (Lemma C.3).

**On the maximally serial case ($\mathrm{WP}_n$ of $A_5$), erasing the chain drops the performance from 1 to the predicted ceiling.** The chain-trained models reach accuracy $1.000$ on $A_5$ at every length up to $n = 128$ (Figure 3a). When their chain is replaced by filler, by random tokens, or by a template, the accuracy drops to between $0.0148$ and $0.0188$ over all lengths against the baseline $1/60 = 0.0167$; the quantitative ceiling predicted by Theorem 3.1 is exactly met. Therefore, the whole accuracy above chance, $1 - b$, rests on the content of the chain.

**After reinforcement learning that rewards only the final answer, erasing the chain still drops the accuracy to the ceiling.** Models trained on $A_5$ at $n = 32$ with a chain and then with GRPO (Shao et al., 2024), RLOO (Ahmadian et al., 2024), or PPO (Schulman et al., 2017) end at $0.709$ to $0.804$ with the chain and at $0.014$ to $0.019$ under every erasing intervention (Figure 3b).

**What survives without a chain is the part of the answer that counting can compute.** At $n = 128$ the chainless models keep exactly the accuracy of computing the commutative part of the product and guessing the rest (Proposition D.2; Figure 3a, blue): all of it on $\mathbb{Z}_{60}$; $1/4$ on $A_4 \times \mathbb{Z}_5$, although its ceiling is 1, because training does not find the shallow circuit for the rest (Appendix H).

**Without a chain, accuracy collapses beyond an input length that grows with the depth, even with supervision at every step.** We follow Li et al. (2024) to train with dense supervision, which provides the model with the running product at every position, so the collapse is not due to insufficient training signal. On $A_5$ the collapse comes at $n = 16$ with $L = 2$ and at $n = 64$ with $L = 4$, while $L = 8$ still holds $0.886$ at $n = 64$ (Figure 3c). By Fact 2.1, a pass of fixed depth (pipeline) performs a bounded number of sequential steps, and for a sufficiently large $n$, no training makes a single pass sufficient.

**One forward pass can perform only a bounded number of sequential steps.** Figure 3d measures this maximum number of steps a forward pass can perform by tracking the prefix $t^*$, the largest position up to which a chainless model still computes the running product correctly. In the maximally serial case ($\mathrm{WP}_n$), $t^*$ stays at about 8 at $L = 2$ for inputs of all lengths, stays at about 30 at $L = 4$, and reaches about 110 at $L = 8$. The only way to extend the bound is a chain, since each generated token adds one more pass (Figure 5).

**A chain helps only when its length grows with the input.** We train models whose chain writes only every $k$-th running product, so the chain has $T = n/k$ steps and each step has to multiply $k$ elements in one pass. Every $T < n$ stays at the baseline and only $T = n$ reaches $1.000$ (Figure 3e).

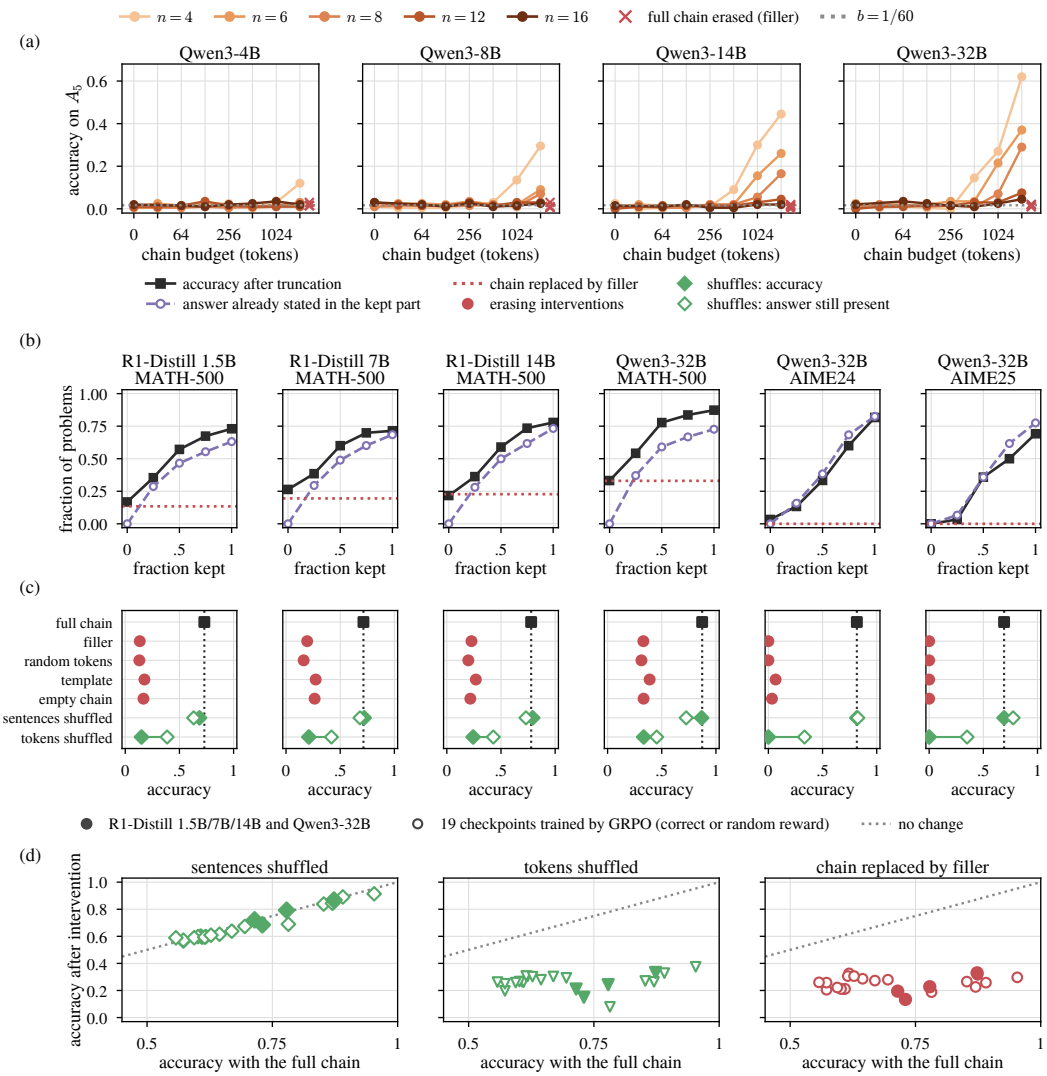


Figure 4: Open-weight reasoning models. *(a)* Qwen3 models on $A_5$ (maximally serial): accuracy after a chain of at most the given number of tokens. *(b)* Mathematical benchmarks: accuracy after truncating the chain to a prefix (solid) against the fraction of problems whose kept prefix already states the answer (dashed). *(c)* Accuracy after each intervention: the four erasing interventions (red) drop the accuracy, a sentence shuffle keeps the accuracy of the full chain, and a token shuffle falls to the level of erasing although some answer is still present in the shuffled chains (hollow diamonds). *(d)* The four untrained models (filled) and 19 checkpoints of Qwen3 models trained by GRPO with a correct or a random reward (hollow), on MATH-500: accuracy after intervention against the accuracy with the full chain (Table 6).

### 6.2 A TASK WITH A KNOWN CEILING AT SCALE: OPEN-WEIGHT REASONING MODELS ON THE WORD PROBLEM

We pose the word problem of $A_5$ to Qwen3 models (Yang et al., 2025) with 4B to 32B parameters in natural language: each element of $A_5$ is given the name of an orientation of an object; the prompt states the starting orientation, the rotations to apply in order, and for each rotation the orientation it produces from every orientation; the model has to name the final orientation, one of 60 ($b = 1/60$). For 200 words at each length $n$ from 4 to 16 we run two protocols: we let the model think for at most $T$ tokens and then force the answer as in Section 6.3, for $T$ from 0 to 2,048; or we let it finish its chain and replace the chain by filler of the same length before forcing the answer (Appendix G).

**Erasing returns the open models to the baseline, and a chain helps only from 512 tokens on.** With its full chain Qwen3-32B reaches 0.62 at $n = 4$ and 0.29 at $n = 8$; with filler, or with no chain at all, every model scores at most 0.030 at every $n$, and equally on $\mathbb{Z}_{60}$ and $A_4 \times \mathbb{Z}_5$ (Figure 4a,

Appendix H). With chains of at most 2,048 tokens, the accuracy falls with $n$, from 0.62 at $n = 4$ to 0.075 at $n = 12$ for Qwen3-32B, as Theorem 4.1 states; beyond $n = 8$ every model scores at most 0.075 even with its full chain.

### 6.3 Tasks with an unknown ceiling: open-weight reasoning models on mathematical benchmarks

To test the implications of Section 3 through Section 5, we apply the interventions to DeepSeek-R1-Distill-Qwen models (Guo et al., 2025) with 1.5B, 7B, and 14B parameters and to Qwen3-32B, on AIME24, AIME25, and MATH-500. Each intervention edits the content of the thinking block, and the model is then forced to state its answer immediately without reopening the block, so everything after the intervened chain is the answer head defined in Section 2. The content-dependent interventions of Section 5 are truncation to a prefix (keeping 0% of the chain gives an empty chain, which is erasing), sentence shuffle, and token shuffle.

**Erasing the chain removes 0.52 to 0.82 accuracy, and what survives is the part that the model can solve without its chain.** Replacing the chain by filler costs the four models 0.52 to 0.60 accuracy on MATH-500 and costs Qwen3-32B 0.82 on AIME24 and 0.69 on AIME25; the four erasing interventions agree within 0.11 (Figure 4c). From Section 3, the surviving accuracy is the part of the benchmark that the model solves without any serial computation. On MATH-500 this floor is 0.13 to 0.33 and is highest for the largest model. The part carried by the chain, 0.52 to 0.60, stays roughly constant across sizes, indicating that a larger model solves more problems in one pass, but the number of problems that need the chain does not go down.

**The model answers correctly in step with how often the kept part of the chain already states the answer.** For Qwen3-32B on AIME24, keeping 25, 50, and 75 percent of the chain gives accuracy 0.133, 0.333, and 0.600, while the fraction of problems whose answer is already *present* in the kept part is 0.158, 0.383, and 0.683 (Figure 4b). By Theorem 5.2, wherever accuracy survives truncation the answer was already readable in the kept prefix. The presence curve shows that the answer enters the chain progressively rather than at the start.

**Shuffling sentences keeps the answer readable while shuffling tokens does not, even though the answer is still there.** Shuffling the sentences of the chain changes accuracy by at most 0.045 on every model and benchmark (Figure 4c). By Theorem 5.2 and Lemma F.1 in Appendix F, this certifies that the answer is shallowly readable from the bag of sentences: the model only needs to select the sentence that states the answer, which is a shallow operation. Shuffling the tokens drops accuracy to the level of an empty chain, 0.000 on AIME and 0.333 on MATH-500, although the answer string is still present in 33% to 45% of the token-shuffled chains. Recovering the answer from a bag of tokens would mean reordering them, which is serial work that a shallow answer head cannot do. The small models show the same bound (Figure 3f).

## 7 Conclusion

In this work, we have shown that a transformer computes each token with one shallow pass, so any procedure that runs it a constant number of times is a shallow circuit. The best accuracy a shallow circuit reaches on a task, the ceiling of the task, bounds the accuracy of every such procedure. We further introduced three theorems that hold for every transformer, no matter how it is trained. Everything a model achieves above the ceiling rests on the content of its chain (Necessity); such a chain cannot be written by a shallow computation, not even approximately (Depth); and the answer is committed by the time the chain ends and is read off by one more shallow pass (Locality). We ran extensive experiments with small transformer models on word problems with known ceilings. We also evaluated open-weight reasoning models on the same word problems, as well as on mathematical benchmarks. Both families of models were tested with and without reinforcement learning. All the results meet what our theorems predict exactly. Our theorems also apply to current systems from one-pass models to diffusion decoders and process rewards (Appendix A). This is a step toward deciding, from the task rather than the model, when a chain of thought is necessary and when a single pass suffices. We hope our work can offer insights for the next harness system that integrates models with varying reasoning capacities for different real-world tasks.

### AI USE STATEMENT

In this work, we used generative AI tools for editing code for the experiments. We have not used generative AI tools for other tasks with required disclosure, and generating synthetic data sets and assisting with translation are not applicable to this work. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by two authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

Leonard M. Adleman. Two theorems on random polynomial time. In *19th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 75–83, 1978.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Diogo Almeida. Introducing System One models & Jev. TypeSafe AI blog, https://typesafe.ai/blog/introducing-system-one-models-and-jev, September 2026. Accessed September 22, 2026.

Alireza Amiri, Xinting Huang, Mark Rofin, and Michael Hahn. Lower bounds for chain-of-thought reasoning in hard-attention transformers. *arXiv preprint arXiv:2502.02393*, 2025.

David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in $NC^1$. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of $NC^1$. *Journal of the ACM*, 35(4):941–952, 1988.

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. Do NOT think that much for 2+3=? On the overthinking of o1-like LLMs. *arXiv preprint arXiv:2412.21187*, 2024.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, Vlad Mikulik, Samuel R. Bowman, Jan Leike, Jared Kaplan, and Ethan Perez. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit CoT to implicit CoT: Learning to internalize CoT step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2502.05171.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.02226.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations (ICLR)*, 2018. arXiv:1711.02281.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Conference on Language Modeling (COLM)*, 2025. arXiv:2412.06769.

Johan Håstad. *Computational Limitations of Small-Depth Circuits*. MIT Press, 1987.

Johan Håstad. On the correlation of parity and small-depth circuits. *SIAM Journal on Computing*, 43(5):1699–1708, 2014.

Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for $AC^0$. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 961–972, 2012.

Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. 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. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2402.12875.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison 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. arXiv:2305.20050.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.10749.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023. arXiv:2207.00729.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.07923.

William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2503.03961.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *International Conference on Machine Learning (ICML)*, 2024.

Eric Miles and Emanuele Viola. Shielding circuits with groups. In *45th ACM Symposium on Theory of Computing (STOC)*, pp. 251–260, 2013. ECCC TR13-003; IACR ePrint 2013/001.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2502.09992.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*, 2024.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2502.17416.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. Spurious rewards: Rethinking training signals in RLVR. *arXiv preprint arXiv:2506.10947*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

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. arXiv:2305.04388.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.

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

## A Discussion: What our theorems say about reasoning models

In this section, we connect our theorem to the capabilities of several recent kinds of reasoning models and discuss what it implies about their performance on different tasks, to provide insights on where and why they can succeed, and when they are expected to fail.

**Chain-free models.** Jev (Almeida, 2026) is a *System One* model (Kahneman, 2011), a one-pass transformer that returns a probability distribution over all possible answers. It can be viewed as a reasoning model with an empty chain, a one-pass pipeline, so Lemma 2.3 applies to quantify its ceiling. Its intended tasks, classification, routing, extraction, and guardrails, are decisions that one pass over the state is expected to make, and in our terms whose ceiling is one. On such tasks, a reasoning model pays $T$ passes for a chain whose erasure costs nothing. On an average-case serial task, however, it fails on an $\varepsilon$ fraction of inputs that no shallow circuit could solve, and on a maximally serial task it is no better than guessing, no matter how it is trained.

**Latent reasoning.** Methods that internalize the chain by training with chains or distilling from a model that writes them (Deng et al., 2023; 2024) share the same bound as the chain-free models, because such a model answers in one pass at test time. Another line of research feeds a hidden vector back in place of a token (Hao et al., 2025) or iterates a block of layers of constant depth at test time (Geiping et al., 2025; Saunshi et al., 2025). A constant number of latent steps is a pipeline and is bounded by the ceiling in the same way (Remark B.5).

**Filler and pause tokens.** Compute without content does not help either. Filler tokens (Pfau et al., 2024) and pause tokens (Goyal et al., 2024) add positions to the context whose content is fixed in advance rather than produced by the model, so the extra computation happens inside one forward pass over a longer context. They add width and not depth, and they recover the benefit of a chain only on parallelizable tasks, whose ceiling is one.

**Non-autoregressive and diffusion decoding.** Non-autoregressive decoding (Gu et al., 2018) and masked diffusion language models (Nie et al., 2025; Inception Labs et al., 2025) unmask a full sequence over a number of denoising steps. That number of steps is fixed in advance and does not grow with the input, so on an average-case serial task they are capped at $1 - \varepsilon$ and on a maximally serial task they are no better than the guessing baseline. If the number of denoising steps grows with the input, the steps take the role of the chain, and Theorem 4.1 applies to the sequence of drafts as it applies to a chain of tokens (Remark B.5).

**Compressing the chain.** Theorem 4.1 says what a model loses when its chain could have been written shallowly, since when $\eta_n$ is close to zero, the model cannot do better than a pipeline. As a result, no training method can compress the reasoning for a serial task into a few tokens, nor into a few latent steps (Remark B.5). The theorem also tells us which part of a chain can be shortened. A model whose whole chain is shallowly written on all but a $\lambda$ fraction of its inputs has success at most $\mathrm{shallow}_n + \lambda$ (Lemma E.1 in Appendix E), so every unit of accuracy above the ceiling is achieved from the input on which the chain is not shallowly written. As a result, compression approaches that prompt the model to produce the fewest possible steps (Xu et al., 2025) or train it to halt after its first correct answer (Chen et al., 2024) can be applied repeatedly until only the steps that rely on earlier steps are left. After that, the accuracy on each input whose chain is further compressed falls to $\mathrm{shallow}_n$, the ceiling of pipeline on this task.

**Process reward models.** Theorem 5.2 tells us which chains can be graded by a shallow process reward. A model that scores a reasoning trace in one pass, such as a process reward model that grades each step (Lightman et al., 2024), is a decoder in the sense of Definition 5.1. Our theorem tells us which chains it can check. It can only check the chain if each step it grades is a shallow consequence of the input and of a bounded number of earlier steps. For example, a running product follows from the previous one and the next input element. In this case, checking the chain means checking each step locally and combining the checks with one AND, which is a shallow computation. When a chain leaves a gap between two steps that only serial work bridges, as the chains of large language models can, no shallow process reward can grade the step that hides that work without redoing it. In the extreme case where a chain provides only its final conclusion, any shallow verifier for that conclusion would also be a shallow solver, so its ability to succeed hinges on whether the problem is solvable by a shallow solver.

## B Details for Section 2: One forward pass and shallow pipelines

**Conventions.** A Boolean circuit is a directed acyclic graph of gates. Its *size* is the number of gates and its *depth* is the length of the longest path from an input to the output. A gate with *unbounded fan-in* reads all $n$ inputs at once. A majority gate outputs 1 whenever strictly more than half of its inputs are 1, which enables counting within a single circuit layer. A *randomized* circuit takes an auxiliary uniformly random string. By Adleman's argument (Adleman, 1978), a randomized circuit family can be turned into a non-uniform deterministic family of the same class if its error on every input is below $2^{-|x|\cdot\omega(1)}$, and in particular below the inverse of the number of inputs. This is because a single random string that is good for every input exists and can be hardwired. We use $\omega(1)$ to denote a quantity that grows without bound as $n \to \infty$, and a function $\mathrm{negl}(n)$ is negligible if it is $n^{-\omega(1)}$, that is, if it decays faster than every inverse polynomial. We write $\mathsf{TC}^0/\mathrm{poly}$ and $\mathsf{AC}^0/\mathrm{poly}$ when we want to stress that a class is non-uniform. Table 1 summarizes the three circuit classes that appear in the paper, and Figure 5 shows the three shapes of computation behind the notion of depth. In the definition of the ceiling in Section 2, the depth and the size have to be fixed before the supremum is taken, because at any single input length a large enough circuit is a lookup table that answers every input, so a supremum over all polynomial-size families would equal 1 at every $n$. We write $\mathrm{shallow}_n$ without the two constants when they are clear from the context, and we read such a statement as follows. An upper bound $A \le \mathrm{shallow}_n$ on the success $A$ of a specific shallow object, such as a pipeline or a decoder, holds with $d$ and $s$ equal to the depth and the size exponent of the circuit of that object. An upper bound $\mathrm{shallow}_n \le B_n$ on the ceiling itself holds for every fixed $d$ and $s$ and for all sufficiently large $n$. The two readings chain, since $A \le \mathrm{shallow}_n(d, s) \le B_n$ for all sufficiently large $n$.

*Remark* B.1 (Sampling conventions). We model the sampling procedure for $\pi$ as a finite-precision program, which takes as input the logits together with a block of random bits and outputs a token.

| Class | Gates | Depth | Contains | Does not contain |
|---|---|---|---|---|
| $\mathsf{AC}^0$ | AND, OR, NOT with unbounded fan-in | $O(1)$ | OR of $n$ bits, addition of two $n$-bit numbers | parity and majority (proved) |
| $\mathsf{TC}^0$ | the same gates and majority | $O(1)$ | counting, sums of $n$ numbers, multiplication, division, sorting, parity, word problems of solvable groups | the word problem of $A_5$ if and only if $\mathsf{TC}^0 \neq \mathsf{NC}^1$ (open) |
| $\mathsf{NC}^1$ | AND, OR, NOT with fan-in two | $O(\log n)$ | the word problem of every finite group, evaluation of Boolean formulas | |

Table 1: The three circuit classes used in this paper. All classes have polynomial size. One forward pass of a transformer lies in $\mathsf{TC}^0$ at log precision and in $\mathsf{AC}^0$ at constant precision (Fact 2.1).

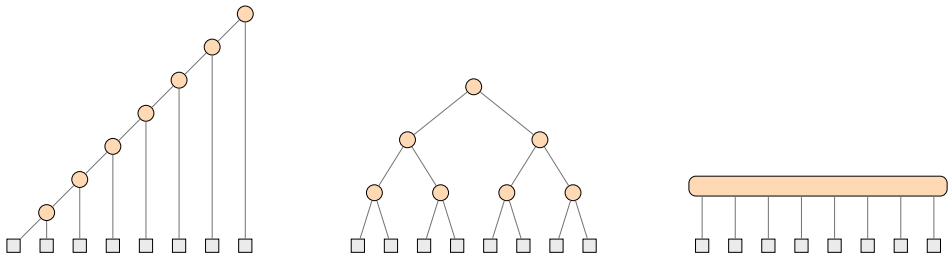


Figure 5: Three circuits that combine $n = 8$ inputs. The sequential chain has depth $n - 1$ and is what a chain of thought implements, one step per token. The balanced tree has depth $\log_2 n$ and is how a circuit in $\mathsf{NC}^1$ multiplies $n$ elements of any finite group. The single wide gate has depth 1 and exists only when the answer depends on counts, which is the case for abelian groups and is the reason their word problems lie in $\mathsf{TC}^0$.

The sampler described in Facts B.2 and B.3 samples exactly from the distribution induced by $\pi$, since they draw by comparing a random value to the cumulative distribution. If instead one wishes to assume ideal sampling from the exact real-valued softmax, the corresponding finite-precision sampler is within total variation distance $\mathrm{negl}(n)$, so all claims in this paper continue to hold with an extra additive $\mathrm{negl}(n)$ term, which is absorbed by the existing slack.

Fact 2.1 is the log-precision case of the following two statements.

**Fact B.2** (Log-precision setting; Merrill & Sabharwal, 2023)**.** *Every function computed by one forward pass of a constant-depth, polynomial-width transformer with $O(\log n)$-bit precision on inputs of length $\mathrm{poly}(n)$ lies in $\mathsf{TC}^0/\mathrm{poly}$. Moreover, sampling one token from the output distribution, given a uniformly random string, is computable by a randomized $\mathsf{TC}^0$ circuit (compute the logits, the cumulative distribution, and a comparison).*

**Fact B.3** (Constant-precision setting; Li et al., 2024, Thm. 3.1)**.** *In the finite-precision model of Li et al. (2024) (constant-bit mantissa and exponent, rounding after each operation), every function computed by one forward pass of a constant-depth transformer lies in $\mathsf{AC}^0/\mathrm{poly}$, and sampling one output token is computable by a randomized $\mathsf{AC}^0$ circuit.*

Precision refers to the number of bits with which the transformer stores each number in its computation, such as an activation or an attention score. A transformer with $O(\log n)$ bits can represent an average over all $n$ positions accurately enough to count, which is what a majority gate does, whereas a transformer with constant precision cannot. This is why the two settings land in $\mathsf{TC}^0$ and in $\mathsf{AC}^0$. The log-precision setting is the standard model in this literature and the one that the main text uses, and we use the word *shallow* for the class that matches the setting.

*Proof of Lemma 2.3.* Fix $n$ and a pipeline with $q$ stages. We show by induction over the stages that the output of every stage is computed by a randomized circuit of polynomial size and constant depth in the class of the setting. The output of every stage is a function of $x$ and of the random bits used so far. For the base case, any stage that uses a shallow randomized map satisfies the claim, since composing it with the circuits from the preceding stages preserves the result. A stage that runs a forward pass consists of three steps. First, it assembles a context of length $\mathrm{poly}(n)$ from $x$ and from the earlier outputs. The layout of the context is fixed in advance, so this step only copies symbols and has constant depth. Second, it runs one forward pass of $\pi$ on this context. The forward pass on a context of length $\mathrm{poly}(n)$ is in $\mathsf{TC}^0$ by Fact B.2 and in $\mathsf{AC}^0$ by Fact B.3, and the softmax with its normalization is part of the cited containments. Third, it samples one token at every selected position. Given the probabilities $(p_1, \ldots, p_{|\Sigma|})$ at the working precision, the circuit computes the prefix sums $s_j = \sum_{i \le j} p_i$, interprets a fresh block of random bits as a number $r' \in [0, 1)$ at the working precision, and outputs the first $j$ with $r' < s_j$. The prefix sums are an iterated addition of polynomially many $O(\log n)$-bit numbers, which is in $\mathsf{TC}^0$, and in the constant-precision setting the alphabet is constant and the additions have constant size, so they are in $\mathsf{AC}^0$. The comparisons and the selection are a constant number of operations over a constant alphabet. The selected positions are handled in parallel by disjoint copies of this sampler with independent random bits, which does not increase the depth. Each stage therefore has polynomial size and constant depth, and composing at most $q = O(1)$ such stages preserves both. The depth of the composed circuit is the sum of the depths of its stages and its size is the sum of their sizes, so the composed circuit lies in $\mathcal{C}_n(d, s)$ for constants $d$ and $s$ that depend only on the depth and the size exponent of the circuits of Facts B.2 and B.3 for $\pi$, on those of the shallow maps of the pipeline, and on $q$. By Remark B.1 the output of the resulting circuit has exactly the distribution of the answer of the pipeline. Its success is the probability, over a random input and the random bits, that its output equals $y(x)$, which is at most $\mathrm{shallow}_n(d, s)$ by the definition of the ceiling in Section 2. □

*Remark* B.4 (Multi-token answers). When the output contains $m = O(1)$ tokens generated autoregressively after the chain, we simply unroll the $m$ successive forward passes. The resulting composite has depth $O(mL) = O(1)$, remains within the stated class, and by Definition 2.2, it just appends $m$ stages to the pipeline. When $m = \omega(1)$, producing the answer introduces its own sequential cost that must be included in $T$, and the theorems then hold with $T + m$ substituted for $T$.

*Remark* B.5 (Latent chains). Neither Fact B.2, Definition 2.2, nor Lemma 2.3 relies on the assumption that the item passed from one forward pass to the next is a token. Assume instead that a pass

outputs a vector containing $\mathrm{poly}(n)$ values, each represented with $O(\log n)$ bits, and that the subsequent pass takes this vector as an input embedding. The function mapping the context to this vector is just the forward pass with the sampling step removed, and therefore it is in $\mathsf{TC}^0$ by Fact B.2. By the same inductive argument, a pipeline whose stages output these vectors forms a shallow circuit, and an erasing intervention that swaps the entire vector sequence for a shallow function of $x$ reduces the system to a two-stage pipeline. Repeating a layer block $r$ times is a special case where the vector is the complete hidden state and the context length does not increase. Hence Theorem 3.1 and Theorem 4.1 also apply when the chain $c$ is replaced by the sequence of latent states, with total variation distances computed over the discretized vectors. Each latent step can encode $\mathrm{poly}(n)$ bits, whereas a token carries only $O(\log |\Sigma|)$ bits; this changes how many steps a task may require (Conjecture E.2) but not whether the required number of steps must scale with $n$.

### B.1 The constant-precision setting and parity

Under constant precision, Fact B.3 implies that a forward pass lies in $\mathsf{AC}^0$, meaning that "shallow" corresponds to $\mathsf{AC}^0$. In this setting, we take parity as an example of inherently serial task according to Definition 2.4.

**Definition B.6** (Parity)**.** In the parity task $\mathrm{PAR}_n$, the input is $x \sim \mathcal{U}(\{0,1\}^n)$, the target is $y(x) = x_1 \oplus \cdots \oplus x_n$, where $\oplus$ denotes addition modulo 2, and the guessing baseline is $b = 1/2$.

**Lemma B.7** (Average-case hardness of parity; unconditional)**.** *There is a constant $\kappa > 0$ such that for every randomized Boolean circuit family of size $s(n) = \mathrm{poly}(n)$ and constant depth $d$, $\Pr_{x\sim\mathcal{U}(\{0,1\}^n),\, r}[C_n(x; r) = \mathrm{PAR}_n(x)] \le \frac{1}{2} + 2^{-\kappa n/(\log s(n))^{d-1}} = \frac{1}{2} + 2^{-n/\mathrm{polylog}(n)}$.*

*Proof.* For deterministic circuits this is the correlation bound of Håstad (2014); see also Impagliazzo et al. (2012) and Håstad (1987). For randomized circuits, the success is an average over the random string of the successes of deterministic circuits, so it is at most their maximum. □

Parity is the right task for this setting because it is hard for $\mathsf{AC}^0$ and easy for $\mathsf{TC}^0$, where a majority gate counts the ones. The word problem of $A_5$ is the right task for the log-precision setting for the mirror-image reason. In the terms of Definition 2.4, Lemma B.7 says that parity on uniform inputs is maximally serial for $\mathsf{AC}^0$, unconditionally, where the passage from a bound for every family to a bound on $\mathrm{shallow}_n(d, s)$ is the one given after Lemma C.3. At log precision parity is not even inherently serial, since a $\mathsf{TC}^0$ circuit computes it on every input. Every result in Sections 2 to 5 is proved from three ingredients only, namely Fact 2.1, Lemma 2.3, and the ceiling bound of Lemma C.3. If we replace them by Fact B.3, the $\mathsf{AC}^0$ case of Lemma 2.3, whose proof above covers both settings, and Lemma B.7, we obtain the constant-precision version of every statement. In that version shallow means $\mathsf{AC}^0$, the task is $\mathrm{PAR}_n$, the guessing baseline is $\frac{1}{2}$, the ceiling is at most $\frac{1}{2} + 2^{-n/\mathrm{polylog}(n)}$, and the conjecture of Appendix C is not needed. The interventions of Section 3 lie in both classes. The two constructions in Appendix F that use counting, Lemma F.1 and Proposition F.2, are stated for the log-precision setting only.

## C Details for Section 6.1: average-case hardness of the word problem of $A_5$

**Fact C.1** (Barrington, 1989; Barrington & Thérien, 1988)**.** *For every fixed finite nonsolvable group $G$, the iterated word problem $\mathrm{WP}_G$ (given $(g_1, \ldots, g_n) \in G^n$, output the product $g_1 g_2 \cdots g_n$) is $\mathsf{NC}^1$-complete under $\mathsf{AC}^0$ (indeed projection) reductions. In particular this holds for $G = A_5$, the alternating group on five elements, which is simple and nonsolvable, with $|A_5| = 60$.*

**The conjecture.** All log-precision claims about $\mathrm{WP}_n$ rely on the standard conjecture $\mathsf{NC}^1 \not\subseteq \mathsf{TC}^0$, which is the sole unproved assumption used in this paper. By Fact C.1, this conjecture is equivalent to $\mathrm{WP}_{A_5} \notin \mathsf{TC}^0/\mathrm{poly}$. Indeed, if there were a constant-depth, polynomial-size family of threshold circuits that computed the product correctly on every input length, then the projection reductions in Fact C.1 would imply $\mathsf{NC}^1 \subseteq \mathsf{TC}^0/\mathrm{poly}$; conversely, $\mathrm{WP}_{A_5}$ is contained in $\mathsf{NC}^1$. Interpreted for a single circuit family, the conjecture asserts that any such family must fail on some length-$n$ input

for infinitely many $n$. The results in this appendix instead assume the almost-everywhere version, which requires the same conclusion for all sufficiently large $n$.

**Assumption C.2** (Almost-everywhere hardness of the word problem of $A_5$)**.** For every polynomial-size threshold circuit family $\{C_n\}$ of constant depth and all sufficiently large $n$, there is an input $x \in A_5^n$ with $C_n(x) \neq y(x)$.

Assumption C.2 is slightly more restrict than the conjecture we use in order to keep the quantifiers tidy. If it were false, then some $\mathsf{NC}^1$-complete problem would admit constant-depth circuits for infinitely many input lengths. This is the version implicitly meant in the main text whenever a bound is stated for $\mathrm{WP}_n$. With the conjecture in its usual formulation, each bound stated below would hold for infinitely many $n$ rather than for all sufficiently large $n$, and that is enough for all of the paper's conceptual takeaways (Appendix C.3).

**Lemma C.3** (Average-case hardness of the word problem of $A_5$; conditional)**.** *Under Assumption C.2, for every polynomial-size randomized* $\mathsf{TC}^0$ *family* $\{C_n\}$ *and every constant* $k$*, for all sufficiently large* $n$*:*

$$\Pr_{x \sim \mathcal{U}(A_5^n),\ r} \big[C_n(x; r) = y(x)\big] \ \leq \ \frac{1}{60} + n^{-k}.$$

Lemma C.3 upper-bounds the best achievable success probability for $\mathrm{WP}_n$ at any fixed depth and size. Suppose $\mathrm{shallow}_n(d, s) > \frac{1}{60} + n^{-k}$ occurred for infinitely many $n$. Then, choosing for each such $n$ a circuit in $\mathcal{C}_n(d, s)$ that succeeds above this threshold, and selecting any circuit of the same depth and size for the remaining values of $n$, would yield a polynomial-size randomized $\mathsf{TC}^0$ family contradicting the lemma. Therefore, for all $d, s$ and any constant $k$, we have $\mathrm{shallow}_n(d, s) \leq \frac{1}{60} + n^{-k}$ for all sufficiently large $n$. Consequently, $\mathrm{WP}_n$ under the uniform distribution is maximally serial as in Definition 2.4(iii), and this is the bound invoked in Section 6.1 for $\mathrm{WP}_n$.

The ceiling in Lemma C.3 is induced from Theorem 3.9 of Miles & Viola (2013). That theorem says that if $\mathsf{TC}^0 \neq \mathsf{NC}^1$, then for every $k$ and for infinitely many $t$, the family of $\mathsf{TC}^0$ circuits of size at most $t^k$ with $k \log t$ output bits is $t^{-k}$-fooled by $(A_5)^t$. A class is said to be $\varepsilon$-fooled if no circuit in it has statistical distance $\varepsilon$ between its output distribution under the uniform distribution on $G^t$ and its output distribution under the uniform distribution on $t$-tuples whose product is $\alpha$, for any choice of $\alpha$. Converting their formulation into the prediction version is just a single averaging argument, which we spell out. Assume a $\mathsf{TC}^0$ family $C$ predicts the product on uniform inputs with success probability $1/m + \delta$. For each $\alpha \in G$, let $D_\alpha$ be the uniform distribution on $n$-tuples with product $\alpha$, set $p_\alpha := \Pr_{x \sim D_\alpha}[C(x) = \alpha]$, and define $q_\alpha := \Pr_{x \sim \mathcal{U}(G^n)}[C(x) = \alpha]$, the overall frequency with which $C$ outputs $\alpha$. Because $\mathcal{U}(G^n)$ is the uniform mixture $\frac{1}{m}\sum_\alpha D_\alpha$, we have $\frac{1}{m}\sum_\alpha p_\alpha = \frac{1}{m} + \delta$, so $\sum_\alpha p_\alpha = 1 + m\delta$, whereas $\sum_\alpha q_\alpha \leq 1$. Therefore, for some $\alpha$ we must have $p_\alpha - q_\alpha \geq \delta$, and for that $\alpha$ the test "output equals $\alpha$" distinguishes $D_\alpha$ from $\mathcal{U}(G^n)$ with advantage $\delta$, contradicting $\delta$-fooling. Their theorem is stated for infinitely many $t$, while our version ranges over all sufficiently large $n$; Appendix C.3 explains how to align these quantifier conventions. The randomization in Step 1 of the proof below is the random self-reduction from their Lemma 3.2, applied here in a two-sided way. The remainder proceeds differently: we use a two-sided self-reduction with profile matching, completed by a covering argument. We retain this approach because it produces the prediction statement directly, and because Remark C.4 extracts that line of reasoning.

*Proof of Lemma C.3.* Throughout, $G = A_5$, $m = |G| = 60$, and $y(x) = g_1 \cdots g_n$ for $x = (g_1, \ldots, g_n) \in G^n$. Suppose, for contradiction, that there is a polynomial-size randomized $\mathsf{TC}^0$ family $\{C_n\}$, a constant $k$, and an infinite set $N \subseteq \mathbb{N}$ such that for all $n \in N$,

$$s_n \ := \ \Pr_{\widetilde{x} \sim \mathcal{U}(G^n),\ \rho} \big[C_n(\widetilde{x}; \rho) = y(\widetilde{x})\big] \ \geq \ \frac{1}{m} + \delta, \qquad \delta := n^{-k}.$$

We construct, for each $n \in N$, a polynomial-size deterministic $\mathsf{TC}^0$ circuit that computes $\mathrm{WP}_n$ exactly on all inputs. Since $N$ is infinite, this contradicts Assumption C.2.

**Step 1: two-sided random self-reduction.** Fix an arbitrary target instance $x = (g_1, \ldots, g_n)$. Draw $h_0, h_1, \ldots, h_n \sim \mathcal{U}(G)$ i.i.d. and define $\widetilde{x} = (\widetilde{x}_1, \ldots, \widetilde{x}_n)$ by $\widetilde{x}_i := h_{i-1}^{-1} g_i h_i$. Telescoping,

$$y(\widetilde{x}) = h_0^{-1} g_1 h_1 \cdot h_1^{-1} g_2 h_2 \cdots h_{n-1}^{-1} g_n h_n = h_0^{-1}\, y(x)\, h_n, \qquad \text{so} \qquad y(x) = h_0\, y(\widetilde{x})\, h_n^{-1}.$$

Define the estimate $\hat{y} := h_0\, C_n(\widetilde{x};\rho)\, h_n^{-1}$ with fresh coins $\rho$.

We use two distributional facts. The first is that $\widetilde{x}$ is uniform and independent of $h_0$. For each fixed value of $h_0$ and the fixed $x$, the map $(h_1,\dots,h_n) \mapsto \widetilde{x}$ is a bijection of $G^n$, since it can be inverted coordinatewise by $h_i = g_i^{-1} h_{i-1} \widetilde{x}_i$, so the conditional law of $\widetilde{x}$ given $h_0$ is uniform for every $h_0$. The second is that the offset is a conjugated copy of an error that does not depend on the instance. Writing $\varepsilon := C_n(\widetilde{x};\rho)\, y(\widetilde{x})^{-1}$,

$$\hat{y}\, y(x)^{-1} = h_0\, C_n(\widetilde{x};\rho)\, h_n^{-1} \cdot \big(h_0\, y(\widetilde{x})\, h_n^{-1}\big)^{-1} = h_0\, C_n(\widetilde{x};\rho)\, y(\widetilde{x})^{-1}\, h_0^{-1} = h_0\, \varepsilon\, h_0^{-1}.$$

By the first fact, $\varepsilon$ is a function of $(\widetilde{x},\rho)$ only and hence independent of $h_0$, and $\varepsilon \sim D$ where $D(g) := \Pr_{\widetilde{x}\sim\mathcal{U},\rho}[\, C_n(\widetilde{x};\rho)\, y(\widetilde{x})^{-1} = g\,]$ is a fixed distribution on $G$ depending only on $(C_n, n)$, with $D(e) = s_n$.

Consequently the offset $\hat{y}\, y(x)^{-1}$ is distributed according to the *conjugation average*

$$\bar{D}(g) \;:=\; \frac{1}{m} \sum_{h\in G} D\big(h^{-1} g\, h\big),$$

which is a class function, that is, $\bar{D}(g) = \bar{D}(wgw^{-1})$ for all $w$, satisfies $\bar{D}(e) = D(e) = s_n$, and does not depend on the target instance $x$. The last property is what the following steps use.

**Step 2: repetition and the empirical profile.** Let $c = c(G) \le m$ be the covering constant of Lemma C.5, set $\gamma := \delta/(16c)$, and repeat Step 1 independently $k^* := \big\lceil 8\gamma^{-2}\big((n{+}2)\ln m{+}\ln 8\big)\big\rceil = O\big(c^2 n/\delta^2\big) = \mathrm{poly}(n)$ times, obtaining estimates $\hat{y}^{(1)},\dots,\hat{y}^{(k^*)}$, i.i.d. with $\hat{y}^{(j)} = e^{(j)} \cdot y(x)$ and $e^{(j)} \sim \bar{D}$. Form the empirical profile $\hat{q}(a) := \frac{1}{k^*}\sum_j \mathbf{1}[\hat{y}^{(j)} = a]$ for each $a \in G$. Since $\mathbb{E}\,\hat{q}(a) = \bar{D}(a\, y(x)^{-1})$, Hoeffding and a union bound over the $m$ atoms give

$$\Pr\Big[\; \exists a :\; \big|\hat{q}(a) - \bar{D}(a\, y(x)^{-1})\big| > \gamma \;\Big] \;\le\; 2m\, e^{-2\gamma^2 k^*} \;\le\; \tfrac{1}{4}\, m^{-(n+1)}.$$

Call the complementary event GOOD.

**Step 3: profile matching with hardwired advice.** The non-uniform advice of the circuit includes a table $\widetilde{D}$ with $|\widetilde{D}(g) - \bar{D}(g)| \le \gamma$ for all $g$. Such a table exists because $\bar{D}$ does not depend on the instance and has $m$ entries, and storing each entry to $O(\log(1/\gamma)) = O(\log n)$ bits suffices. The decoder outputs

$$v^\star \;\in\; \arg\min_{v\in G}\; \Delta(v), \qquad \Delta(v) := \max_{a\in G} \big|\hat{q}(a) - \widetilde{D}(a\, v^{-1})\big|,$$

breaking ties arbitrarily. On GOOD, $\Delta(y(x)) \le 2\gamma$, since the sampling error is at most $\gamma$ and the table error is at most $\gamma$.

**Step 4: simplicity forces uniqueness of the match.** Suppose that on GOOD the minimizer were some $v \ne y(x)$. Then $\Delta(v) \le \Delta(y(x)) \le 2\gamma$, hence $|\hat{q}(a) - \bar{D}(a\, v^{-1})| \le 3\gamma$ for all $a$ after removing the table error, and the triangle inequality against $|\hat{q}(a) - \bar{D}(a\, y(x)^{-1})| \le \gamma$ gives, for all $a \in G$,

$$\big|\bar{D}(a\, y(x)^{-1}) - \bar{D}(a\, v^{-1})\big| \le 4\gamma.$$

Substituting $a = b\, y(x)$ and setting $u := y(x)\, v^{-1} \ne e$: for all $b \in G$,

$$\big|\bar{D}(b) - \bar{D}(b\, u)\big| \le 4\gamma. \tag{1}$$

We now extend equation 1 from $u$ to all of $G$ in three steps. The first step extends it to $u^{-1}$. Substituting $b \mapsto bu^{-1}$ in equation 1 gives $|\bar{D}(bu^{-1}) - \bar{D}(b)| \le 4\gamma$ for all $b$. The second step extends it to every conjugate $w = huh^{-1}$, and likewise to $hu^{-1}h^{-1}$. Since $\bar{D}$ is a class function,

$$\bar{D}(b\, huh^{-1}) = \bar{D}\big(h^{-1} b\, h\, u\big), \qquad \bar{D}(b) = \bar{D}\big(h^{-1} b\, h\big),$$

so $|\bar{D}(bw) - \bar{D}(b)| = |\bar{D}(b'u) - \bar{D}(b')| \le 4\gamma$ with $b' = h^{-1}bh$. Thus equation 1 holds with any element of the set $K$ of Lemma C.5 in place of $u$. The third step extends it to an arbitrary $g \in G$. Writing $g = w_1 \cdots w_\ell$ with $w_j \in K$ and $\ell \le c$ by Lemma C.5 and telescoping,

$$\big|\bar{D}(b\, g) - \bar{D}(b)\big| \;\le\; \sum_{j=1}^{\ell} \big|\bar{D}(b\, w_1 \cdots w_j) - \bar{D}(b\, w_1 \cdots w_{j-1})\big| \;\le\; 4c\gamma \qquad \text{for all } b, g \in G.$$

Taking $b = e$ shows $\bar{D}$ is within $4c\gamma$ of the constant $\bar{D}(e)$ pointwise; summing over $G$ and using $\sum_g \bar{D}(g) = 1$,

$$\left|\bar{D}(e) - \tfrac{1}{m}\right| \;\le\; 4c\gamma \;=\; \frac{\delta}{4}.$$

But $\bar{D}(e) = s_n \ge \frac{1}{m} + \delta$, a contradiction. Hence on GOOD the unique minimizer is $v^\star = y(x)$, and the reduction errs with probability at most $\frac{1}{4}m^{-(n+1)}$ *on every input* $x$.

**Step 5: complexity of the wrapper and derandomization.** The wrapper implementing the $k^*$ oracle invocations consists of five components. The coordinate-wise products $h_{i-1}^{-1} g_i h_i$ can be computed by table lookups on triples of symbols from a 60-symbol alphabet, and therefore fall in $\mathsf{NC}^0$; the same holds for the final multiplications $h_0(\cdot)h_n^{-1}$. The $k^*$ instances of $C_n$ are executed in parallel, each having polynomial size and identical constant depth. For the $m$ atoms, the quantities $k^*\hat{q}(a)$ are obtained via iterated addition of $k^*$ bits using majority gates, placing this part in $\mathsf{TC}^0$. Comparing $O(\log n)$-bit fixed-point values to the hardwired table, as well as forming the $m^2$ absolute differences $|\hat{q}(a) - \bar{D}(av^{-1})|$, can be done in $\mathsf{AC}^0$. Taking the argmin over the $m = 60$ possibilities is merely a constant-size selection step. Altogether, the resulting circuit has constant depth and polynomial size, hence is in $\mathsf{TC}^0$, and it uses randomness from $(h_i^{(j)})_{i,j}$ together with the oracle's random bits. For any fixed input its error is at most $\frac{1}{4}m^{-(n+1)} < m^{-n}$. As there are exactly $m^n$ inputs, a union bound implies that a uniformly random choice of the random string works simultaneously for all inputs with probability at least $3/4$. Therefore a good string exists, and fixing it in the circuit (as in Adleman's method) gives a deterministic $\mathsf{TC}^0/\mathrm{poly}$ family computing $\mathrm{WP}_n$ exactly for every $n \in N$. By Fact C.1, this would put an $\mathsf{NC}^1$-complete problem in $\mathsf{TC}^0/\mathrm{poly}$ for infinitely many input lengths, contradicting Assumption C.2. □

*Remark* C.4 (Where simplicity enters, and where $S_5$ fails). The only place where the proof uses the structure of $G$ is Step 4. The passage from near-invariance under one $u \neq e$ to near-invariance under all of $G$ requires that the conjugates of $u^{\pm 1}$ generate $G$, that is, that the normal closure of every nontrivial element is the whole group, which is exactly simplicity. For $G = S_5$ and $u \in A_5$ the normal closure is $A_5 \neq S_5$, so the telescoping argument only forces $\bar{D}$ to be constant on the cosets of $A_5$, and no contradiction arises at $s = 2/|S_5| = 1/60$. This is the correct behavior, because Proposition D.1 exhibits a $\mathsf{TC}^0$ circuit that reaches exactly this value by computing the sign of the product.

### C.1 THE COVERING LEMMA

**Lemma C.5** (Covering in finite simple groups)**.** *Let $G$ be a finite nonabelian simple group and $u \in G \setminus \{e\}$. Let $K = \{gug^{-1} : g \in G\} \cup \{gu^{-1}g^{-1} : g \in G\}$. Then there exists $c = c(G) \le |G|$ such that every element of $G$ is a product of at most $c$ elements of $K$.*

*Proof.* The set $K$ is symmetric, since the inverse of a conjugate of $u$ is the same conjugate of $u^{-1}$, and it is closed under conjugation by construction. Let $H := \langle K \rangle$. Since conjugation permutes $K$, it maps $H$ to itself, so $H \trianglelefteq G$, and $H \neq \{e\}$ since $u \in K$. By simplicity, $H = G$. Now let $S_j := \{w_1 \cdots w_i : 0 \le i \le j,\ w_1, \ldots, w_i \in K\}$ denote the set of products of at most $j$ factors from $K$, with the empty product giving $e \in S_0$. The chain $S_0 \subseteq S_1 \subseteq \cdots$ is nondecreasing. If $S_{j+1} = S_j$ for some $j$, then $S_j K \subseteq S_j$, and by induction $S_j$ contains every finite product of elements of $K$. Since $K$ is symmetric and $G$ is finite, the set of such products is the subgroup $\langle K \rangle = G$, so $S_j = G$. Each strict inclusion in the chain adds at least one element of the finite group, so stabilization, and hence $S_j = G$, occurs at some $j \le |G|$. Taking $c := |G|$ proves the claim. The known covering numbers of $A_5$ are far smaller, but the proof of Lemma C.3 only needs some constant, so we keep the self-contained argument. □

### C.2 THE WORD PROBLEM AT EACH DEGREE

The example of Section 6.1 places the word problem at each of the three degrees of Definition 2.4, depending on the group and on the input distribution. We collect the four statements and their proofs here. Throughout this subsection shallow means $\mathsf{TC}^0$, the class of the log-precision setting.

For a commutative group, such as $\mathbb{Z}_{60}$, the cyclic group of the integers modulo 60, a shallow circuit computes the product on every input, so the ceiling is 1 and the task is not serial in any sense. For $A_5$ (the 60 even permutations of five objects) under a distribution that puts almost all of its mass on inputs with few non-identity elements, no shallow circuit computes the product on every input, so the task is inherently serial, but a shallow circuit solves every input with few non-identity elements, which carry almost all of the mass, so the ceiling tends to 1 (Proposition C.6). For $S_5$, the symmetric group of all 120 permutations of five objects, a shallow circuit is able to compute the sign of the product by counting the number of odd factors with majority gates, and guesses the rest, so the ceiling is $2b = 1/60$ and the task is average-case serial. For $A_5$ on uniform inputs, no shallow circuit predicts the product better than guessing (Miles & Viola, 2013), so the ceiling is $b = 1/60$ and the task is maximally serial.

**Solvable groups: no degree.** If $G$ is solvable, and in particular if it is commutative, then $\mathrm{WP}_G$ lies in $\mathsf{TC}^0$ (Barrington & Thérien, 1988), so a shallow circuit computes the product on every input, the ceiling is one under every input distribution, and none of the three conditions holds. This covers the two control groups $\mathbb{Z}_{60}$ and $A_4 \times \mathbb{Z}_5$ of Section 6.

**$S_5$ on uniform inputs: average-case serial, not maximally serial.** The sign of a product is the parity of the signs of its factors, which a $\mathsf{TC}^0$ circuit computes by counting the odd permutations, and a circuit that outputs a fixed element of the correct coset of $A_5$ is right with probability $1/|A_5| = 1/60 = 2/|S_5|$ (Proposition D.1, proved in Appendix D). Under Assumption C.2, no polynomial-size randomized $\mathsf{TC}^0$ family exceeds $1/60 + n^{-k}$ on uniform inputs from $S_5^n$, by the variant of the proof of Lemma C.3 given in the same appendix. Hence $\mathrm{shallow}_n \le 1/60 + n^{-k}$, which is bounded away from one, so the task is average-case serial with $\varepsilon = 1 - 1/60 - o(1)$; and $\mathrm{shallow}_n \ge 1/60 = 2b$, so it is not maximally serial.

**$A_5$ on uniform inputs: maximally serial.** This is Lemma C.3.

**$A_5$ under a skewed distribution: inherently serial, not average-case serial.** The first degree does not depend on the input distribution, and the second does. The following proposition makes the difference concrete on the same group.

**Proposition C.6** (The distribution decides the degree)**.** *Let $E_n \subseteq A_5^n$ be the set of inputs with at most three entries different from the identity, and let $D_n$ be any distribution on $A_5^n$ with $D_n(E_n) \ge 1 - 2^{-n}$. A deterministic $\mathsf{TC}^0$ family computes the product on every input in $E_n$, so the ceiling of $\mathrm{WP}_{A_5}$ under $D_n$ is at least $1 - 2^{-n}$ and the task is not average-case serial, whereas under Assumption C.2 it remains inherently serial.*

*Proof.* For each position $i$ the circuit computes the bit $z_i := \mathbf{1}[g_i \neq e]$ by a lookup and the prefix count $c_i := \sum_{j \le i} z_j$, an iterated addition of $n$ bits, which lies in $\mathsf{TC}^0$. For $j \in \{1, 2, 3\}$ it forms the one-hot encoding of the $j$-th non-identity entry as the bitwise OR, over all positions $i$, of the encoding of $g_i$ gated by $\mathbf{1}[z_i = 1 \wedge c_i = j]$. On an input in $E_n$ at most one position passes each gate, so the OR returns exactly that entry, and an all-zero result is read as $e$. The circuit then outputs the product of the three retrieved entries, a lookup on three symbols. On every input in $E_n$ this product equals $y(x)$, because the identity entries contribute nothing to $g_1 \cdots g_n$. The circuit has polynomial size and constant depth, so its accuracy under $D_n$ is at least $D_n(E_n) \ge 1 - 2^{-n}$, and the ceiling is at least that; it is not bounded away from one, so Definition 2.4(ii) fails. Definition 2.4(i) does not involve the distribution, and Assumption C.2 says exactly that every $\mathsf{TC}^0$ family errs on some input of $A_5^n$ for all sufficiently large $n$, so the task is inherently serial. The number three is not essential, and the same construction handles any constant number of non-identity entries. □

### C.3 On almost-everywhere versus infinitely-often hardness

Assumption C.2 is stated almost everywhere so that Lemma C.3 holds for all sufficiently large $n$. Under the conjecture in its standard form, which by Fact C.1 says that $\mathrm{WP}_{A_5} \notin \mathsf{TC}^0/\mathrm{poly}$, that is, that every family fails on infinitely many $n$, the same proof shows that the bound of Lemma C.3 holds for infinitely many $n$. The results of Sections 3 to 5 then hold along that subsequence, which suffices for every qualitative conclusion of this paper, since the experiments operate at fixed $n$ in any case. We use the almost-everywhere form because it reads more easily.

## D PROOFS FOR SECTION 3

**Formal setting.** We follow the conventions from Appendix B. An intervention is a family $\Phi = \{\Phi_n\}$ of randomized maps $\Phi_n : \Sigma^n \times \Sigma^{T(n)} \to \Sigma^{T'(n)}$ with $T'(n) \le \mathrm{poly}(n)$, such that $\Phi_n$ can be computed from $(x, c)$ together with the intervention's random string by a constant-depth, polynomial-size randomized circuit in the relevant class for the setting. The specific interventions we study—constants, truncation, independent random tokens, and shuffling—belong to both classes. We call an intervention erasing if $\Phi_n(x, c) = \varphi_n(x; r)$ for some randomized map $\varphi_n$ in the same class and some random string $r$. Figure 6 depicts the protocol along with these two types of interventions.

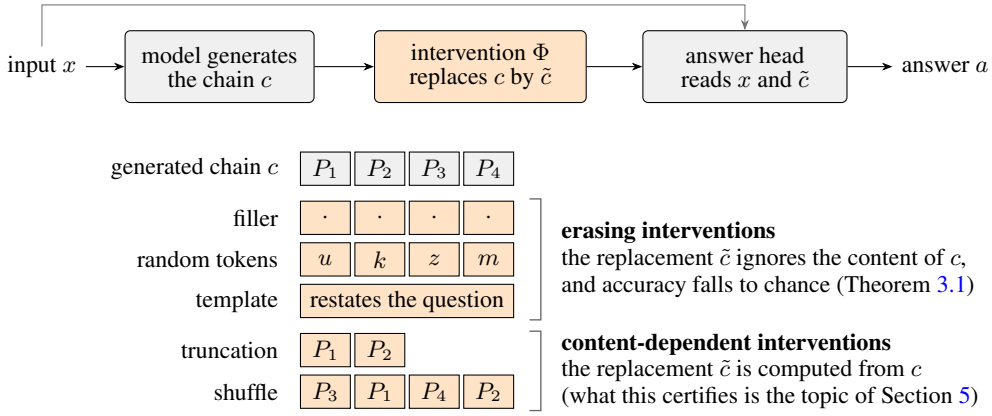


Figure 6: Test-time interventions. The model generates its chain $c$ as usual. The intervention $\Phi$ then replaces $c$ by a modified chain $\tilde{c}$, and the answer head produces the answer from the input and $\tilde{c}$. The load-bearing gap is the accuracy that the model loses in this way. The lower part shows examples of $\tilde{c}$ (orange) for a chain that consists of four running products $P_1, \ldots, P_4$.

*Proof of Theorem 3.1.* Fix $n$ and an erasing intervention with $\tilde{c} = \varphi_n(x; r)$. The intervened model runs $\pi$ for $T$ passes to generate a chain $c$, exactly as the reasoning model does, but an erasing intervention discards $c$, so the model then samples $a \sim \pi(\cdot \mid x, \tilde{c})$ from a context that depends on $(x, \tilde{c})$ alone. Neither the law of $\tilde{c}$ nor the conditional law of $a$ given $(x, \tilde{c})$ depends on $c$, so the $T$ passes that produced $c$ contribute nothing to the distribution of the answer, and marginalizing over $c$ leaves

$$\mathrm{succ}_n(\pi^{\Phi}) \;=\; \Pr_{x,\, r,\, r'} \big[\, A_n(x, \varphi_n(x; r); r') = y(x) \,\big],$$

where $A_n(x, \tilde{c}\,; r')$ denotes one forward pass of $\pi$ on the context formed by $x$, a separator, and $\tilde{c}$, which has length $\mathrm{poly}(n)$, followed by sampling one token from the answer head with the random string $r'$. The right-hand side is the success of a pipeline with two stages in the sense of Definition 2.2. The first stage applies the shallow map $\varphi_n$, and the second stage runs the forward pass and samples the answer. The intervened model is therefore not a pipeline as a procedure, since it performs $T(n)$ serial passes, but its answer has the same distribution as the answer of a two-stage pipeline, which is all that Lemma 2.3 needs. The lemma bounds $\mathrm{succ}_n(\pi^{\Phi})$ by $\mathrm{shallow}_n(d, s)$ for constants $d$ and $s$ that depend only on the forward pass of $\pi$ and on $\varphi_n$, hence only on $\pi$ and $\Phi$, and the bound on the gap follows from its definition. On $\mathrm{WP}_n$, Lemma C.3 bounds $\mathrm{shallow}_n(d, s)$ by $\frac{1}{60} + n^{-k}$ for every constant $k$ and all sufficiently large $n$, so every shallow imitator of a model with success $p$ on $\mathrm{WP}_n$ has error at least $p - \frac{1}{60} - n^{-k}$. On the other hand, a shallowly written chain caps the accuracy on $\mathrm{WP}_n$ at $\frac{1}{60} + n^{-k}$. □

**Shortcuts.** On a maximally serial task no part of the answer can be computed without serial work. On other tasks some part can, and we call such a part a shortcut. A shortcut narrows the answer down to a few candidates, and guessing among them makes it easier for a shallow circuit to get the correct answer.

**Proposition D.1** (Shortcuts raise the ceiling)**.** *Suppose that some feature $\phi(y(x))$ of the answer can be computed from the input $x$ by a shallow randomized circuit family $\phi$, and knowing this value*

*leaves at most $m$ candidate answers. Then* $\mathrm{shallow}_n \geq 1/m$*, and a shallow circuit that computes the feature and guesses among the candidates reaches accuracy* $1/m$ *without any chain.*

*Proof of Proposition D.1.* Let $A$ be the shallow randomized family that computes the feature, so $A(x; r) = \phi(y(x))$ for every input. For each value $q$ in the range of $\phi$, fix an answer $a^*(q)$ that maximizes $\Pr_x[\, y(x) = a \mid \phi(y(x)) = q \,]$ over the answers $a$. Since at most $m$ answers are consistent with $q$, this maximum is at least $1/m$. The circuit computes $q = A(x; r)$ and outputs $a^*(q)$ by a lookup table with polynomially many entries, which is a circuit of constant depth. Its accuracy is $\sum_q \Pr[\phi(y(x)) = q] \cdot \Pr[y(x) = a^*(q) \mid \phi(y(x)) = q] \geq 1/m$. The table can be hardwired because our circuit families are non-uniform. The circuit lies in $\mathcal{C}_n(d, s)$ for the depth and the size exponent of $A$ increased by those of the lookup, so $\mathrm{shallow}_n(d, s) \geq 1/m$ for these constants, and since the circuit reads only $x$ it reaches this accuracy without any chain, which gives the last statement. □

**Shortcuts through a quotient.** In the case of word problems, the answer feature described in Proposition D.1 is most naturally obtained by taking an appropriate quotient of the group. The formulation below applies to all finite groups, and in the special case of abelianization it gives the expression $1/|[G, G]|$. Table 2 reports the resulting ceilings for the groups studied in this paper as well as for the two control groups introduced in Section 6: the cyclic group $\mathbb{Z}_{60}$ of integers modulo 60 and the solvable group $A_4 \times \mathbb{Z}_5$. Both control groups have 60 elements, like $A_5$, and therefore share the same vocabulary and the same guessing baseline.

**Proposition D.2** (Shortcuts through a quotient)**.** *Let $G$ be a finite group and let $\phi : G \to Q$ be a surjective homomorphism onto a group $Q$ whose word problem lies in $\mathsf{TC}^0$. This holds for every commutative group $Q$ and, by Barrington & Thérien (1988), for every solvable group $Q$. Then, in the log-precision setting, a shallow circuit family reaches accuracy $1/|\ker \phi|$ on the word problem of $G$.*

*Proof.* The circuit applies $\phi$ to every input element, which is a lookup per position. It then computes $q = \phi(g_1) \cdots \phi(g_n) = \phi(y(x))$ with a $\mathsf{TC}^0$ circuit for the word problem of $Q$, and it outputs a fixed element of the fiber $\phi^{-1}(q)$. For uniform inputs the product $y(x)$ is uniform on $G$, so conditioned on $\phi(y(x)) = q$ it is uniform on the fiber, which has $|\ker \phi|$ elements. The output is therefore correct with probability $1/|\ker \phi|$. □

| Group $G$ | $\lvert G\rvert$ | $[G, G]$ | chance $1/\lvert G\rvert$ | price of the abelian quotient $1/\lvert[G, G]\rvert$ | best shallow success (ceiling) |
|---|---|---|---|---|---|
| $\mathbb{Z}_{60}$ (abelian) | 60 | $\{e\}$ | $1/60$ | 1 | 1 |
| $A_4 \times \mathbb{Z}_5$ (solvable) | 60 | $V_4$ | $1/60$ | $1/4$ | 1 |
| $S_5$ | 120 | $A_5$ | $1/120$ | $1/60$ | $1/60 + o(1)$ |
| $A_5$ (simple) | 60 | $A_5$ | $1/60$ | $1/60$ | $1/60 + o(1)$ |

Table 2: What a shallow circuit can obtain on the word problem of each group. The price of the abelian quotient is the success of the circuit that computes the image of the product in $G/[G, G]$ and guesses inside the coset (Proposition D.2). The last column is the ceiling $\mathrm{shallow}_n(d, s)$ at every fixed depth and size, i.e., the best success of a shallow circuit, up to the error term $n^{-k}$. It equals 1 for the two solvable groups, because their word problems lie in $\mathsf{TC}^0$; these two groups are the controls of Section 6. For $S_5$ and $A_5$ the last column holds under Assumption C.2 (Proposition D.1 and Lemma C.3). The ceiling equals the guessing baseline only when the group has no shallow quotient, which is why Section 6.1 instantiates Theorem 3.1 on $A_5$.

For $G = S_5$ with the sign homomorphism, the sign of a product is determined by the parity of the signs of the individual factors, the kernel is $A_5$, and the resulting accuracy is $1/60 = 2/|S_5|$; this matches the lower bound for the $S_5$ case of Proposition D.1 in Section 6.1. Miles & Viola (2013) already point to the sign map on $S_5$ as the motivation for working over $A_5$. Proposition D.2 additionally quantifies the cost, translating this obstruction into an attainable accuracy level for a predictor. Proposition D.2 implies that enforcing a ceiling at the pure-guessing rate $1/|G|$ demands that $G$ have no nontrivial quotient whose word problem is in $\mathsf{TC}^0$. It is enough that $G$ be perfect: any quotient of a perfect group remains perfect, any nontrivial perfect group is not solvable, and the

word problem for a nonsolvable group is $\mathsf{NC}^1$-complete (Barrington & Thérien, 1988). Simplicity is a stronger condition used by our proof of Lemma C.3, and $A_5$ satisfies both properties. For the two solvable control groups, we can take $\phi$ to be the identity, so a shallow circuit can compute the full product and the ceiling becomes 1.

*Proof of the upper bound for $S_5$ in the instantiation of Proposition D.1.* We follow the proof of Lemma C.3 with $G = S_5$ and $m = |G| = 120$, and we describe only the changes. Suppose that a polynomial-size randomized $\mathsf{TC}^0$ family $\{C_n\}$ has success $s_n \geq \frac{1}{60} + \delta$ with $\delta = n^{-k}$ on uniform inputs from $S_5^n$, for all $n$ in an infinite set $N$. Steps 1 to 3 of that proof use only the group axioms, so they carry over unchanged with $\gamma := \delta/(16c)$, where $c \leq |G|$ is the constant from the covering argument below. They yield the class function $\bar{D}$ with $\bar{D}(e) = s_n$ and the profile-matching decoder.

In Step 4, suppose that on the event GOOD the minimizer is some $v \neq y(x)$, and let $u := y(x)\,v^{-1} \neq e$. As before, $|\bar{D}(b) - \bar{D}(bw)| \leq 4\gamma$ holds for all $b \in G$ and for every $w$ in the set $K$ of conjugates of $u$ and of $u^{-1}$. Let $H := \langle K \rangle$, which is a normal subgroup of $G$. The proof of Lemma C.5 uses simplicity only to conclude that $H = G$. Without this hypothesis, the same argument shows that every element of $H$ is a product of at most $c \leq |G|$ elements of $K$. Telescoping as before gives $|\bar{D}(g) - \bar{D}(e)| \leq 4c\gamma$ for all $g \in H$. The only normal subgroups of $S_5$ are $\{e\}$, $A_5$, and $S_5$, so $A_5 \subseteq H$. Summing over $g \in A_5$ yields

$$60\,\big(\bar{D}(e) - 4c\gamma\big) \;\leq\; \sum_{g \in A_5} \bar{D}(g) \;\leq\; 1,$$

and hence $\bar{D}(e) \leq \frac{1}{60} + 4c\gamma = \frac{1}{60} + \frac{\delta}{4}$, which contradicts $\bar{D}(e) = s_n \geq \frac{1}{60} + \delta$. On GOOD the unique minimizer is therefore $y(x)$.

Step 5 then yields a deterministic $\mathsf{TC}^0/\mathrm{poly}$ family that computes the word problem of $S_5$ exactly for every $n \in N$. Since $A_5$ is a subgroup of $S_5$, the same family computes $\mathrm{WP}_n$ on inputs from $A_5^n$, which contradicts Assumption C.2. □

## E Proofs for Section 4

**Formal setting.** Given two distributions $P$ and $Q$ over the same finite domain, their total variation distance is defined as $\mathrm{TV}(P, Q) := \max_A |P(A) - Q(A)|$, where the maximum is taken over all events $A$. Let $\pi_c(\cdot \mid x)$ denote the distribution over chains produced by $\pi$'s reasoning model on input $x$. A shallow imitator is a shallow randomized family $\varphi$ that, from an input $x$ together with its internal randomness, outputs a chain; we write $\varphi(x)$ for the resulting output distribution on $x$, and define its imitation error as $\eta_n := \mathbb{E}_x\ \mathrm{TV}\big(\varphi(x),\, \pi_c(\cdot \mid x)\big)$. We will use the standard fact that for any function $h : [0, 1]$-valued, $|\mathbb{E}_P h - \mathbb{E}_Q h| \leq \mathrm{TV}(P, Q)$.

*Proof of Theorem 4.1.* The idea is that a shallow imitator is itself an erasing intervention, so the theorem reduces to Theorem 3.1 through a total variation argument. Let $\varphi$ be a shallow imitator of $\pi$ with imitation error $\eta_n$, and use it as the erasing intervention $\Phi(x, c) := \varphi(x)$. For a fixed input $x$, let $h_x(c) := \pi\big(y(x) \mid x, c\big) \in [0, 1]$ be the probability that the answer head outputs the target when it reads the chain $c$. The success of the model on $x$ is $\mathbb{E}_{c \sim \pi_c(\cdot|x)}\, h_x(c)$, and the success of $\pi^\Phi$ on $x$ is $\mathbb{E}_{\tilde{c} \sim \varphi(x)}\, h_x(\tilde{c})$. By the property above, the two differ by at most $\mathrm{TV}(\varphi(x), \pi_c(\cdot \mid x))$. Averaging over $x$ gives $\mathrm{succ}_n(\pi^\Phi) \geq \mathrm{succ}_n(\pi) - \eta_n$. Theorem 3.1 bounds the left-hand side by $\mathrm{shallow}_n(d, s)$ for constants $d$ and $s$ that depend only on $\pi$ and $\varphi$, so $\eta_n \geq \mathrm{succ}_n(\pi) - \mathrm{shallow}_n(d, s)$, which is the claim. On $\mathrm{WP}_n$, Lemma C.3 bounds $\mathrm{shallow}_n(d, s)$ by $\frac{1}{60} + n^{-k}$ for every constant $k$ and all sufficiently large $n$, which gives the instantiation stated in Section 6.1. □

**Shallowly written chains.** Constant-length chains and drafts polished for only a constant number of rounds (Section 4) can be viewed as chains generated by a pipeline as in Definition 2.2. Concretely, for a chain of fixed length $T$ the pipeline performs one forward pass per token, and for a draft updated over $q = O(1)$ rounds it performs one forward pass per round. In round $j$, the model runs a forward pass on the input together with the tokens produced in rounds $1, \ldots, j-1$, and then samples tokens at some chosen positions. Masked diffusion decoding with $q$ denoising steps fits this second template: at step $j$ it runs a forward pass on the partially unmasked sequence obtained from

steps $1, \dots, j-1$ and samples tokens exactly at the positions it reveals. By Lemma 2.3, the output distribution of any such pipeline matches that of a shallow randomized circuit, so it is a shallow imitator with $\eta_n = 0$; likewise, a chain produced by a shallow map is, by definition, a shallow imitator with $\eta_n = 0$. In each of these three settings, Theorem 4.1 yields $\mathrm{succ}_n(\pi) \le \mathrm{shallow}_n$.

For the mixture case, write $\pi_c(\cdot \mid x) = (1 - \lambda(x))\,\mu_x + \lambda(x)\,\nu_x$, where $\mu_x$ is the output distribution of a shallow map and $\nu_x$ is arbitrary. Then $\mathrm{TV}(\mu_x, \pi_c(\cdot \mid x)) = \lambda(x)\ \mathrm{TV}(\mu_x, \nu_x) \le \lambda(x)$, so the shallow map serves as an imitator with $\eta_n \le \mathbb{E}_x \lambda(x) = \lambda$. Applying Theorem 4.1 gives $\mathrm{succ}_n(\pi) \le \mathrm{shallow}_n + \lambda$; taking $\lambda$ to be constant in the statement corresponds to the special case where $\lambda(x)$ is constant.

**Lemma E.1** (Boilerplate does not enter the bound)**.** *Suppose the chain of a reasoning model splits as $c = (c_D, c_S)$, where $c_S = f(x, c_D)$ for a shallow randomized map $f$, and write $\pi_{c_D}(\cdot \mid x)$ for the distribution of $c_D$. Then for every shallow imitator $\psi$ of $c_D$, $\mathrm{succ}_n(\pi) \le \mathrm{shallow}_n + \mathbb{E}_x\ \mathrm{TV}\big(\psi(x), \pi_{c_D}(\cdot \mid x)\big)$; that is, Theorem 4.1 holds with $c_D$ in place of $c$.*

*Proof.* Define $\varphi(x)$ by drawing $c'_D \sim \psi(x)$ and outputting $(c'_D, f(x, c'_D))$. This is a shallow imitator of $c$, since it composes two shallow randomized maps. The chain of the model is $(c_D, f(x, c_D))$ with $c_D \sim \pi_{c_D}(\cdot \mid x)$, so $\varphi(x)$ and $\pi_c(\cdot \mid x)$ are the images of $\psi(x)$ and $\pi_{c_D}(\cdot \mid x)$ under the same kernel $c_D \mapsto (c_D, f(x, c_D))$, and applying one kernel to two distributions does not increase their total variation distance. Hence $\mathrm{TV}(\varphi(x), \pi_c(\cdot \mid x)) \le \mathrm{TV}(\psi(x), \pi_{c_D}(\cdot \mid x))$ for every $x$, and Theorem 4.1 applied to $\varphi$ gives the claim. □

**Conjecture E.2** (Linear length)**.** *On $\mathrm{WP}_n$, every log-precision transformer whose success tends to 1 requires a chain of length $T(n) = \Omega(n/\log n)$.*

The analogous statement with $T(n) = \Omega(n)$ is a theorem for parity in the unique-hard-attention regime (Amiri et al., 2025). We leave the version for soft attention and log precision open, which is an instance of Open Problem 5.1 of Amiri et al. (2025). Theorem 4.1 shows that the number of sequential steps has to grow without bound. The conjecture concerns the rate.

## F Proofs for Section 5

**Formal setting.** For a transformer $\pi$, an intervention $\Phi$, and a task with target $y$, the shallow decodability of Definition 5.1 is

$$\mathrm{dec}_\Phi(\pi; d, s) \;:=\; \sup_{D \in \mathcal{C}_n(d,s)} \Pr_{x,\ c \sim \pi_c(\cdot|x),\ r, r'} \Big[ D\big(x,\ \Phi_n(x, c; r);\ r'\big) = y(x) \Big],$$

where the supremum ranges over randomized decoders $D$ of depth at most $d$ and size at most $n^s$ in the class of the setting.

*Proof of Theorem 5.2.* The intervened answer is drawn as $a \sim \pi(\cdot \mid x, \tilde{c})$. By Fact B.2, Fact B.3, and the sampling construction of Appendix B, the map $(x, \tilde{c}; r') \mapsto a$ is a randomized circuit of constant depth and polynomial size in the class of the setting, with a depth $d$ and a size exponent $s$ that depend only on $\pi$. It is therefore an admissible decoder $D_0 \in \mathcal{C}_n(d, s)$, and $\Pr[D_0(x, \Phi(x, c; r); r') = y(x)] = \mathrm{succ}_n(\pi^\Phi)$ by the definition of the intervened model. Hence $\mathrm{succ}_n(\pi^\Phi) \le \mathrm{dec}_\Phi(\pi; d, s)$. This argument uses neither the task nor the input distribution. For an erasing intervention, a decoder in $\mathcal{C}_n(d, s)$ applied to $(x, \varphi_n(x; r))$ is a pipeline with two stages and hence a shallow randomized circuit (Lemma 2.3), whose depth and size exponent $d'$ and $s'$ depend only on $d$, $s$, and $\varphi_n$, so $\mathrm{dec}_\Phi(\pi; d, s) \le \mathrm{shallow}_n(d', s')$; in particular a decoder that ignores the chain has accuracy at most $\mathrm{shallow}_n$, whereas $\mathrm{dec}(\pi) \ge \mathrm{succ}_n(\pi) = p$ by the first statement. Combined with the first statement, the erasing bound recovers Theorem 3.1. On $\mathrm{WP}_n$, no decoder reads the answer from an erased chain with accuracy above $\frac{1}{60} + n^{-k}$ (Lemma C.3), whereas the answer head of a model with success $p$ reads it from the model's own chain with accuracy $p$; the chain raises the shallow decodability from chance to $p$. □

**Shuffling and the bag.** Fix a way to segment the chain into units—either tokens or sentences, where sentences are split at sentence-final punctuation and line breaks. This segmentation can be computed by a shallow circuit, since it merely needs to identify the delimiters. The shuffle intervention $\Phi^{\mathrm{shuf}}$ returns the units of $c$ arranged according to a uniformly random permutation $\sigma$. Let

$\mathrm{bag}(c)$ denote the multiset of units in $c$, and define $\mathrm{dec}_{\mathrm{bag}}(\pi)$ as the intervention that produces a canonical encoding of $\mathrm{bag}(c)$, e.g., by listing the units in sorted order.

**Lemma F.1** (Shuffle sees the bag)**.** *For either segmentation, in the log-precision setting, for every $d$ and $s$ there are constants $d'$ and $s'$ such that* $\mathrm{dec}_{\Phi^{\mathrm{shuf}}}(\pi; d, s) - \mathrm{negl}(n) \leq \mathrm{dec}_{\mathrm{bag}}(\pi; d', s')$ *and* $\mathrm{dec}_{\mathrm{bag}}(\pi; d, s) \leq \mathrm{dec}_{\Phi^{\mathrm{shuf}}}(\pi; d', s')$.

*Proof.* We first show the second inequality. Given $\sigma(c)$, the canonical sorted sequence is a deterministic $\mathsf{TC}^0$ computation that depends only on the bag, since the ranks follow from iterated addition and comparisons, so any bag decoder in $\mathcal{C}_n(d, s)$ becomes a shuffle decoder with identical success, at a constant additional depth and a polynomial additional size, which fixes $d'$ and $s'$. We then show the first inequality. Given the bag, assign each unit an independent random key of $C \log n$ bits and order the units by key. Conditioned on all keys being distinct, which fails with probability at most $T^2 2^{-C \log n} = \mathrm{negl}(n)$ for a large constant $C$, the resulting ordering is exactly uniform, that is, it has the law of $\sigma(c)$ given the bag. Running any shuffle decoder in $\mathcal{C}_n(d, s)$ on this simulated input loses at most the failure probability, again at a constant additional depth and a polynomial additional size. □

Together with Theorem 5.2, the lemma gives $\mathrm{succ}_n(\pi^{\Phi^{\mathrm{shuf}}}) \leq \mathrm{dec}_{\mathrm{bag}}(\pi) + \mathrm{negl}(n)$, which is the statement about shuffling in Section 5.

**Proposition F.2** (Fragile to erasure and robust to shuffling)**.** *There is a polynomial-size log-precision transformer $\pi^\star$ with $\mathrm{succ}_n(\pi^\star) = 1$ on $\mathrm{WP}_n$ whose load-bearing gap is 0 under shuffling. Under Assumption C.2, its gap is at least $1 - \frac{1}{60} - n^{-k}$ under every erasing intervention, for every constant $k$ and all sufficiently large $n$.*

*Proof.* On input $x$, the transformer $\pi^\star$ deterministically emits $c = (P_1, \ldots, P_n, y, y, \ldots, y)$, where $P_t = g_1 \cdots g_t$ are the prefix products and the final block repeats $P_n = y(x)$ exactly $2n + 1$ times, so $T = 3n + 1$. Its answer head outputs the plurality element of the chain-token sub-vocabulary in its context. Such a generator exists, because computing $P_t$ from $(P_{t-1}, g_t)$ is a single multiplication in a finite group, which one attention step implements by retrieving the previous chain token and the aligned input token, followed by a constant lookup. This is a special case of the step simulation of Li et al. (2024, Thm. 3.3), and repeating the last value is trivial. Such an answer head exists as well. With one-hot value vectors, uniform attention over the chain segment yields the exact frequency vector of the 60 symbols, where segment embeddings distinguish chain tokens from input tokens, and each entry is a multiple of $1/T$ and representable at $O(\log n)$ bits. Uniform attention scores do not depend on the order of the tokens, which is exactly why the head is robust to shuffling. The plurality symbol leads every other symbol by a margin of at least $(n + 1)/(3n + 1) > 1/3$, so a constant-size feed-forward argmax with constant separation suffices and tolerates the rounding. The transformer is correct on its own chain and on every permutation of it. In the unshuffled chain, and in any permutation of it, the multiplicity of $y(x)$ is at least $2n + 1$, while any other symbol occurs at most $n$ times among the prefixes, so the plurality is $y(x)$ deterministically, $\mathrm{succ}_n(\pi^\star) = \mathrm{succ}_n((\pi^\star)^{\Phi^{\mathrm{shuf}}}) = 1$, and the gap under shuffling is 0. Finally, $\pi^\star$ is fragile to erasure. It is a polynomial-size log-precision transformer, so Theorem 3.1 applies to it as stated, and Lemma C.3 bounds the ceiling of $\mathrm{WP}_n$. For every erasing intervention, the intervened success is therefore at most $\frac{1}{60} + n^{-k}$. □

**Fragility under content-dependent interventions.** Theorem 5.2 limits what the answer head can contribute, but it does not restrict how the answer head acts on chains the model would never generate on its own, such as a permuted or truncated chain. Consequently, a reduction in accuracy under such an intervention reflects the answer head rather than the chain itself, while an erasing intervention guarantees a drop for any model by Theorem 3.1. The next proposition highlights this contrast in its most extreme form. Section 6 leverages it to interpret the interventions on open models.

**Proposition F.3** (Fragility does not mean that the answer is absent)**.** *There are two transformers on $\mathrm{WP}_n$ that generate exactly the same chains and both have success 1. The first keeps success 1 when its chain is shuffled. The second falls to $\frac{1}{60} + O(1/n)$ under the same intervention, although a decoder reads the answer from its shuffled chain with accuracy 1.*

*Proof of Proposition F.3.* The first transformer is $\pi^\star$ from Proposition F.2. Let $\pi'$ agree with $\pi^\star$ on generation (hence identical chains and identical success 1) but differ in the behavior of the answer head on contexts outside the support of generation. On any context whose chain segment is not of the canonical form, that is, prefix products followed by a constant block, $\pi'$ outputs a fixed symbol $z \in G$. Both behaviors are realizable by polynomial-size log-precision transformers. The two heads are gated by a constant-depth check of the canonical form, which verifies $c_t = c_{t-1}\, g_t$ for $t \le n$ with the convention $c_0 := e$, and $c_t = c_{t-1}$ for $t > n$, and this check is an $\mathsf{NC}^0$ test per position aggregated by an AND. Under $\Phi^{\mathrm{shuf}}$, passing the canonical-form check pins down the whole sequence, because the check forces $c_1 = g_1$ and each subsequent token by induction, so a shuffled chain passes if and only if the permutation reproduces the canonical sequence exactly. For fixed $x$, this happens with probability $\prod_g m_g! \,/\, (3n+1)!$ under a uniform permutation, where $m_g$ are the multiplicities of the chain's token values. If the bag contains at least two distinct values, $\prod_g m_g! \le m_{\max}!\,(3n+1-m_{\max})!$, so the probability is at most $1/\binom{3n+1}{m_{\max}} \le 1/(3n+1)$; the bag is single-valued only on the degenerate instances $x = (y, e, \ldots, e)$, a $60^{1-n}$ fraction of inputs. Hence $\mathrm{succ}_n(\pi'^{\Phi^{\mathrm{shuf}}}) \le \frac{1}{60} + \frac{1}{3n+1} + 60^{1-n} = \frac{1}{60} + O(1/n)$, while $\mathrm{succ}_n((\pi^\star)^{\Phi^{\mathrm{shuf}}}) = 1$. The answer head of $\pi^\star$ is a decoder that reads the answer from the shuffled chain of $\pi'$ with accuracy 1, because the two transformers generate the same chains. □

| Intervention | The answer head still sees | If the accuracy stays high, we learn that | If the accuracy drops, we learn that |
|---|---|---|---|
| Erasing (filler, random tokens, template, empty chain) | nothing from the chain | the model can answer these inputs without its chain, so a shortcut exists | the content of the chain was needed. On a maximally serial task the drop is forced and goes all the way to the guessing baseline (Theorem 3.1) |
| Truncation (keep a prefix) | the first part of the chain | the answer was already determined at that point of the chain | this answer head cannot finish the remaining work in one pass. We learn nothing about the chain itself |
| Shuffle (tokens or sentences) | the bag of tokens or of sentences | the answer is readable from the bag, so the chain states it in a way that does not depend on order | nothing about the chain. Two policies with identical chains can react in opposite ways (Proposition F.3) |
| Re-encoding (relabeling, paraphrase) | all of the information, in a different form | the answer head relies on features that survive the re-encoding | nothing about the chain. The information is unchanged, so the drop only reflects the answer head |

Table 3: How to read an intervention experiment. By Theorem 5.2, the accuracy after an intervention is bounded by what a shallow decoder can read from the part of the chain that the intervention keeps. High accuracy is therefore informative for every intervention. A drop in accuracy is informative about the chain only for erasing interventions.

## G Experimental setup

**Small transformers.** We train small transformer models with $L \in \{2, 4, 8\}$ layers at input lengths $n$ ranging from 8 to 256 on four word problems. The tasks are $\mathrm{WP}_n$, the word problem of $A_5$; the word problem of $S_5$ (Proposition D.1 in Appendix D); and the word problems of the two solvable control groups $\mathbb{Z}_{60}$ and $A_4 \times \mathbb{Z}_5$ of Table 2 in Appendix D, which have ceiling 1 and the same vocabulary and baseline as $A_5$. Two training regimes are applied to each task. The first trains the model to generate the chain of running products defined in Section 6.1 before answering. The second trains the model to answer in a single forward pass without generating a chain. A chainless model is a pipeline with one forward pass, so Lemma 2.3 caps its success at the ceiling of its task, which is listed in Table 2 for the four groups.

All models are decoder-only transformers with rotary positional embeddings, hidden size 512, 8 attention heads, and $L \in \{2, 4, 8\}$ layers. They are trained from scratch with batch size 256 for 6,000 steps when $n \le 32$, 10,000 steps when $n \in \{64, 128\}$, and 16,000 steps when $n = 256$. Inputs

consist of uniformly sampled group-element sequences, represented with one token per element, and evaluation is performed on 10,000 new inputs using a fixed evaluation seed. In the chain-trained setting, the model is trained to output the running products $P_t = g_1 \cdots g_t$ as an explicit chain and then produce the final answer; erasing interventions in this setting replace that chain with filler tokens, uniformly random tokens, or a fixed template. In the no-chain setting, the model outputs the answer in a single forward pass (thus forming a one-pass pipeline under Definition 2.2) and is trained with the correct running product as the target at every position, providing the strongest supervision available without an explicit chain. Parity experiments follow Definition B.6, and control groups are as specified in Table 2. Most core cells use three random seeds, while the $S_5$ cells at $n \in \{32, 64, 128\}$ use eight. For each run, the tracked prefix $t^*$ is defined as the largest index $t$ such that per-position accuracy is at least 0.9 for every position up through $t$. At $n = 256$, the fixed training budget becomes insufficient even for commutative controls (whose maximum accuracy is 1), leading to reduced accuracy; accordingly, plateau comparisons in the main text focus on $n \leq 128$. The no-chain runs were introduced after the main program was finalized and were labeled retrospectively in the released logs.

**Chain formats, relabeling, and the bag decoder.** For Figure 3f, models with $L = 4$ are trained on $A_5$ at $n \in \{16, 32, 64\}$ with three seeds on one of three chains: the running products $P_1, \ldots, P_n$; the running products with each product written twice; and the running products followed by $2n + 1$ copies of $P_n = y(x)$, the chain of Proposition F.2. Besides the three erasing interventions, we apply a token shuffle, which permutes the chain tokens uniformly at random, and a relabeling, which applies one fixed random bijection of the group to every chain token. The bag decoder is the better of a logistic regression and a network with one hidden layer of 128 units, both fit on the vector that counts how often each group element occurs in the chain the model generated, on 70% of the evaluation inputs, and scored on the remaining 30%. Its accuracy is a lower bound on the bag decodability of Lemma F.1.

**Chain length.** The chain that writes every $k$-th running product lists $P_k, P_{2k}, \ldots$ and always ends with $P_n$, so it has $T = \lceil n/k \rceil$ steps and its last token is the answer; $T = n$ is the full chain. Models with $L = 4$ are trained at $n \in \{16, 32, 64, 128\}$ for every $T \in \{1, 2, 4, \ldots, n\}$, with three seeds each.

**Reinforcement learning on the word problem.** Models with $L = 4$ are trained on $A_5$ at $n = 32$ on the chain for 3,000 supervised steps, which leaves their accuracy well below one, and then for 1,500 steps with GRPO, RLOO, or PPO, with learning rate $10^{-4}$, a KL coefficient of 0.03, and reward 1 for a correct final answer and 0 otherwise. We first ran RL after training on the chain to accuracy 1; every sample then received reward 1, the policy gradient vanished, and RL left the model unchanged, so we report only the protocol above.

**Open reasoning models.** We test DeepSeek-R1-Distill-Qwen-1.5B, -7B, and -14B, as well as Qwen3-32B, on MATH-500 and additionally evaluate Qwen3-32B on AIME24 and AIME25, using four generations per problem and seed 1234 for both the engine and the sampler. Checkpoints and datasets were downloaded from the Hugging Face hub without pinning specific revisions. The released records specify the chat templates used to format the prompts, and the latest hub modification of those templates predates our executions. The model initially produces its solution within a think block, sampled with temperature 0.6, top-$p$ 0.95, and top-$k$ 20, using a chain budget of 16,384 tokens for models with at least 14B parameters and 8,192 tokens for smaller models. Our intervention overwrites the think-block content, then closes the block, after which decoding proceeds greedily under a forced suffix ending in `Final Answer: \boxed{`, for up to 64 tokens, while a logit bias prevents the think tag from being generated so the block cannot be reopened. Consequently, everything generated after the intervened chain corresponds to the answer head in Section 2, i.e., a single forward pass on the input plus the intervened chain; Remark B.4 discusses the case where the answer spans multiple tokens.

We define six channels. *Filler* substitutes the chain with a neutral token sequence of the same length. *Random tokens* swaps it for uniformly sampled tokens with matched length. *Template* repeats the question text until it reaches the chain's length. *Sentence shuffle* permutes the chain's sentences uniformly at random: the chain is segmented at sentence-final punctuation and line breaks, and each sentence remains unchanged. *Token shuffle* randomly permutes all tokens in the chain. *Truncation* retains the specified fraction of the chain's tokens. The first three are erasing as defined in Section 3,

while the last three are content-dependent as defined in Section 5; Table 3 describes what each one certifies. For each problem, we also note whether the ground-truth answer appears verbatim anywhere in the intervened chain (referred to in the main text as the answer being present); for randomized channels, this is evaluated on an independent sample from the same channel. Accuracy is computed as exact match on the forced answer. The released records have two caveats. We did not store the generated chain texts, only aggregated per-cell metrics, so quantities that require the raw chains (e.g., sentences per chain) cannot be recomputed. Additionally, for the three distilled models whose chat template ends the prompt with an open think tag followed by a newline, the intervened context re-adds the tag but omits the newline. This occurs consistently across all channels, including the identity condition, so within-cell comparisons remain consistent, but each intervened context differs from the clean run by that single character.

**Open models on the word problem.** We render each word of $\mathrm{WP}_n$ in natural language. The 60 elements of $A_5$ receive fixed names of orientations of an object, such as *Upright-North*. The prompt states that the object starts in the orientation named by the identity, lists the rotations $g_1, \ldots, g_n$ by name, and gives, for every rotation that occurs in the word, one line with the orientation reached from each orientation, that is, the corresponding column of the multiplication table. The answer is the final orientation, so $b = 1/60$. The controls $\mathbb{Z}_{60}$ and $A_4 \times \mathbb{Z}_5$ use the same names and the same template, so only the multiplication table changes. We also include parity, rendered as a coin that is flipped or left alone at each step ($b = 1/2$), and a retrieval task that counts marked items in a context whose length grows with $n$. We use the post-trained Qwen3 models with 4B, 8B, 14B, and 32B parameters in thinking mode, 200 words per task and length, and the sampling settings above with seed 1234, with at most 2,048 chain tokens. For the budget sweep, the model generates at most $T \in \{0, 32, 64, \ldots, 2{,}048\}$ thinking tokens, the block is closed, and the answer is forced with the suffix `The final orientation is:` for up to 24 tokens, with the think tag banned; $T = 0$ closes the block immediately. For the erasing intervention, the full chain is replaced by the same number of copies of one neutral token, and the accuracy with the full chain is measured with the same forced suffix.

**Reinforcement-learning checkpoints.** We train Qwen3 models with GRPO (Shao et al., 2024) in verl (Sheng et al., 2024), with a token-level loss, on mathematical problems from DeepScaleR and from the harder part of DeepMath, after removing every problem that shares a word $n$-gram with MATH-500, AIME24, or AIME25. The reward is 1 for a correct final answer, or a pseudo-random bit that does not depend on the answer (Shao et al., 2025). The correct reward is applied to the post-trained Qwen3-4B, 8B, and 14B, and the random reward to the post-trained Qwen3-4B and 8B and to Qwen3-4B-Base and Qwen3-8B-Base; one checkpoint of Qwen3-4B is trained with a reward for the answer format only. Table 6 lists the 22 checkpoints we evaluated and their training steps. They are evaluated on MATH-500 with the protocol above, on 500 problems with four generations each, except for four checkpoints evaluated on 200 problems and two on 64 problems with two generations. On three checkpoints the forced answer ignores the chain: every intervention gives the same accuracy, and the answer is never present in the chain. We exclude these three from Figure 4d.

## H Additional experimental results

**What chainless models learn.** Figure 7 (left and middle) looks inside the chainless models of Section 6.1. On $A_4 \times \mathbb{Z}_5$, six reruns at $n = 64$ and 128 predict the coset of the running product modulo $V_4 = [G, G]$ with accuracy 0.999 to 1.000, averaged over positions $t \geq 16$, while their accuracy on the product itself is 0.249 to 0.251. They compute the image of the product in the abelian quotient and guess inside the coset, which is the circuit of Proposition D.2, and they never find the shallow circuit for the rest of the product. On $S_5$, we count a seed as having learned the sign when its sign accuracy at the answer is at least 0.95 and its accuracy lies within three standard errors of $1/60$, and as not having learned it when its sign accuracy is at most 0.55 and its accuracy is at most 0.012. At $n = 16$, 32, 64, 128, and 256, the sign is learned in 2 of 3, 6 of 8, 5 of 8, 8 of 8, and 0 of 3 seeds, and one seed at $n = 64$ is in between. Seeds that learn it score 0.0146 to 0.0200, and the others score 0.0068 to 0.0096. A hand-coded shallow circuit that computes the sign as the parity of the signs of the factors and guesses inside the coset reaches 0.016 to 0.021 at $n = 8$ to 64, the value $2b$ of Proposition D.2, while a uniform guess on $A_5$ reaches 0.014 to 0.021. With the final answer as the only target, training does not find the sign at $n = 16$ to 64: a model trained to predict only the sign stays at 0.48 to 0.50, and so do a linear probe for the sign on the hidden state of a

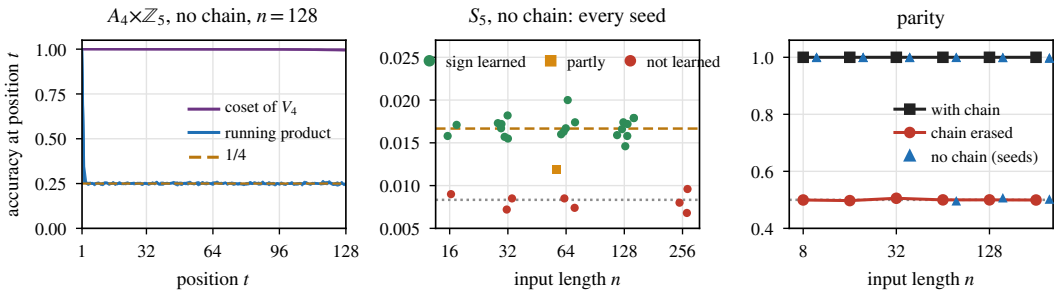


Figure 7: What small transformers learn without a chain, and parity ($L = 4$). *Left:* on $A_4 \times \mathbb{Z}_5$ at $n = 128$, chainless models predict the coset of the running product modulo $V_4 = [G, G]$ at every position and are right about the product itself a quarter of the time (three seeds). *Middle:* every chainless seed on $S_5$; seeds that learn the sign of the product land at the ceiling $2b = 1/60$ (dashed), the others at $b = 1/120$ (dotted). *Right:* on parity, chain-trained models fall to $1/2$ when their chain is erased, while chainless models with dense supervision (one triangle per seed) solve it in 15 of 18 seeds.

chainless model (0.47 to 0.53), training for 120,000 steps (0.46 to 0.51), and pretraining on the sign before training on the product (0.48 to 0.52). The sign is a parity of the input, and only the running targets of dense supervision let training find it.

**Parity.** Parity (Definition B.6) is maximally serial at constant precision (Lemma B.7) but not serial at log precision, where a majority gate computes it (Appendix B.1). Chain-trained models reach 1.000 at every length from 8 to 256 and fall to between 0.493 and 0.509 when their chain is erased. Chainless models with dense supervision reach at least 0.997 in 15 of 18 seeds and stay at 0.50 in the other three (Figure 7, right), consistent with the log-precision setting. As on the two non-serial groups, the drop under erasing comes from the answer head of the chain-trained models and not from the task.

**Models trained on the answer alone or with filler tokens.** Besides the chain and dense supervision, we trained models that answer immediately with the final answer as the only target, and models whose chain positions hold a fixed filler token. On all four groups and at every length, both stay within 0.0012 of the baseline in the mean over seeds, including $\mathbb{Z}_{60}$ at $n = 8$, which chainless models with dense supervision solve; on parity they reach 0.66 at $n = 8$ and 0.50 beyond. These regimes learn nothing, so they neither support nor test the theorems, and we report them only for completeness.

**Witnesses and twins.** Table 4 realizes Propositions F.2 and F.3. A hand-coded answer head that outputs the most frequent chain token, run on the chain of Proposition F.2, keeps 1.000 under a token shuffle and falls to the baseline or below under erasing. Models trained on this chain survive the shuffle across 12 architectures (0.690 to 1.000 at $n = 32$, with hidden size 256 or 512, $L \in \{2, 4, 8\}$, and learned or rotary positions) and at every length from 18 to 56 (0.696 to 0.926), always below the trained bag decoder. For Proposition F.3, two hand-coded heads read identical chains: the head that takes the plurality keeps 1.000 under a shuffle, and the head gated on the canonical form of the chain falls to 0.015 to 0.018, although the plurality head reads the answer from the same shuffled chains. Two students cloned from the chains of one trained teacher, one with the teacher's architecture and one with eight layers, react to a shuffle in the same way (0.032 to 0.077, below the bag decoder at 0.034 to 0.108), so training does not by itself produce twins that diverge.

**Open models on the word problem and its controls.** Table 5 gives the full chain and the erased chain for every model, task, and length. Replacing the chain by filler drops every model to the baseline on $A_5$, and equally on $\mathbb{Z}_{60}$ and $A_4 \times \mathbb{Z}_5$, whose ceiling is 1, while with the full chain the three groups are solved about equally often. The models compute all three products by stepping through the word and do not use the shortcut that the solvable groups admit. On parity the chain helps every model except Qwen3-8B, and filler returns them to about $1/2$; on the retrieval task, filler costs Qwen3-32B 0.02 at $n = 4$ and 0.80 at $n = 16$. Before running this experiment we expected the drop under filler to be near zero on the solvable groups and on the retrieval task, and largest on

Table 4: Witnesses for Propositions F.2 and F.3 on the word problem of $A_5$. Ranges over input lengths, architectures, and the three erasing interventions (means over three seeds, or over three evaluation seeds for the hand-coded heads). The bag decoder is trained on the counts of the chain tokens; the hand-coded twins read identical chains, so the plurality head is itself a decoder with accuracy 1 on the shuffled chains of the gated head.

| setting | $n$ | with chain | chain erased | tokens shuffled | bag decoder |
|---|---|---|---|---|---|
| Hand-coded plurality head on the chain of Prop. F.2 | 16–64 | 1.000 | 0.000–0.018 | 1.000 | 1.000 |
| Trained on that chain, 12 architectures | 32 | 1.000 | 0.015–0.019 | 0.690–1.000 | 1.000 |
| Trained on that chain, $L = 4$ | 18–56 | 1.000 | 0.014–0.019 | 0.696–0.926 | 1.000 |
| Twin with plurality head (Prop. F.3) | 16–64 | 1.000 | 0.000–0.021 | 1.000 | – |
| Twin with gated head (Prop. F.3) | 16–64 | 1.000 | 0.015–0.018 | 0.015–0.018 | – |
| Student cloned from a trained teacher, same architecture | 16–64 | 1.000 | 0.009–0.019 | 0.032–0.077 | 0.034–0.108 |
| Student cloned from the same teacher, $L = 8$ | 16–64 | 0.999–1.000 | 0.013–0.019 | 0.032–0.077 | 0.034–0.108 |

Table 5: Open models on the word problem of $A_5$ and on its controls, 200 problems per cell: accuracy with the full chain / with the chain replaced by filler of the same length. The baseline is $1/60$ on the three groups and $1/2$ on parity; the retrieval task counts marked items in a context that grows with $n$.

| task | model | $n = 4$ | $n = 6$ | $n = 8$ | $n = 16$ |
|---|---|---|---|---|---|
| $A_5$ | Qwen3-4B | 0.12 / 0.01 | 0.03 / 0.01 | 0.03 / 0.01 | 0.02 / 0.03 |
| | Qwen3-8B | 0.29 / 0.01 | 0.09 / 0.03 | 0.07 / 0.01 | 0.03 / 0.01 |
| | Qwen3-14B | 0.45 / 0.02 | 0.26 / 0.01 | 0.17 / 0.00 | 0.02 / 0.01 |
| | Qwen3-32B | 0.62 / 0.01 | 0.37 / 0.02 | 0.29 / 0.01 | 0.04 / 0.02 |
| $\mathbb{Z}_{60}$ | Qwen3-4B | 0.14 / 0.03 | 0.03 / 0.01 | 0.02 / 0.01 | 0.01 / 0.01 |
| | Qwen3-8B | 0.35 / 0.03 | 0.12 / 0.01 | 0.04 / 0.03 | 0.02 / 0.01 |
| | Qwen3-14B | 0.47 / 0.01 | 0.23 / 0.01 | 0.07 / 0.03 | 0.01 / 0.01 |
| | Qwen3-32B | 0.60 / 0.01 | 0.32 / 0.02 | 0.14 / 0.01 | 0.01 / 0.01 |
| $A_4 \times \mathbb{Z}_5$ | Qwen3-4B | 0.12 / 0.01 | 0.04 / 0.03 | 0.04 / 0.03 | 0.03 / 0.01 |
| | Qwen3-8B | 0.34 / 0.03 | 0.15 / 0.03 | 0.04 / 0.04 | 0.03 / 0.01 |
| | Qwen3-14B | 0.52 / 0.01 | 0.30 / 0.03 | 0.10 / 0.01 | 0.02 / 0.01 |
| | Qwen3-32B | 0.65 / 0.03 | 0.45 / 0.03 | 0.24 / 0.01 | 0.04 / 0.03 |
| parity | Qwen3-4B | 0.58 / 0.46 | 0.76 / 0.49 | 0.81 / 0.47 | 0.70 / 0.49 |
| | Qwen3-8B | 0.47 / 0.46 | 0.49 / 0.49 | 0.47 / 0.47 | 0.56 / 0.49 |
| | Qwen3-14B | 0.99 / 0.46 | 0.95 / 0.49 | 0.93 / 0.47 | 0.51 / 0.49 |
| | Qwen3-32B | 0.64 / 0.56 | 0.73 / 0.51 | 0.72 / 0.44 | 0.91 / 0.48 |
| retrieval | Qwen3-4B | 0.82 / 0.39 | 0.81 / 0.32 | 0.66 / 0.29 | 0.62 / 0.13 |
| | Qwen3-8B | 0.81 / 0.34 | 0.97 / 0.48 | 0.99 / 0.34 | 0.98 / 0.13 |
| | Qwen3-14B | 1.00 / 0.56 | 0.99 / 0.46 | 0.96 / 0.45 | 0.97 / 0.20 |
| | Qwen3-32B | 0.99 / 0.97 | 0.99 / 0.65 | 0.97 / 0.38 | 0.97 / 0.17 |

$A_5$ and parity. That expectation did not hold. The theorems bound the accuracy after erasing from above and say nothing about how far a model that relies on its chain falls when the ceiling is one.

**Reinforcement-learning checkpoints.** Table 6 lists the accuracy of every checkpoint of Figure 4d under every intervention, and the three checkpoints we excluded.

Table 6: The 22 Qwen3 checkpoints trained by GRPO that we evaluated on MATH-500 (records are problems times generations): accuracy with the full chain and after each intervention, all with the forced answer. † Excluded from Figure 4d: the forced answer of these checkpoints ignores the chain, so every intervention gives the same accuracy, and the answer is never present in the chain.

| start model | reward | step | records | full chain | filler | random | template | empty | sent. shuffle | token shuffle |
|---|---|---|---|---|---|---|---|---|---|---|
| Qwen3-4B | correct | 25 | 128 | 0.891 | 0.258 | 0.266 | 0.297 | 0.297 | 0.891 | 0.328 |
| Qwen3-4B | correct | 50 | 128 | 0.953 | 0.297 | 0.273 | 0.336 | 0.312 | 0.914 | 0.375 |
| Qwen3-4B | correct | 425 | 2000 | 0.573 | 0.205 | 0.206 | 0.242 | 0.218 | 0.566 | 0.197 |
| Qwen3-8B | correct | 45 | 2000 | 0.853 | 0.266 | 0.244 | 0.296 | 0.304 | 0.839 | 0.271 |
| Qwen3-8B | correct | 255 | 2000 | 0.782 | 0.188 | 0.157 | 0.268 | 0.274 | 0.690 | 0.079 |
| Qwen3-14B | correct | 110 | 2000 | 0.874 | 0.316 | 0.300 | 0.334 | 0.340 | 0.861 | 0.332 |
| Qwen3-4B† | format only | 750 | 800 | 0.010 | 0.010 | 0.010 | 0.010 | 0.010 | 0.010 | 0.010 |
| Qwen3-4B | random | 25 | 800 | 0.870 | 0.226 | 0.225 | 0.276 | 0.280 | 0.844 | 0.264 |
| Qwen3-4B† | random | 475 | 800 | 0.020 | 0.020 | 0.020 | 0.020 | 0.020 | 0.020 | 0.020 |
| Qwen3-4B-Base | random | 25 | 2000 | 0.573 | 0.258 | 0.262 | 0.286 | 0.274 | 0.572 | 0.249 |
| Qwen3-4B-Base | random | 50 | 2000 | 0.557 | 0.260 | 0.254 | 0.289 | 0.266 | 0.589 | 0.262 |
| Qwen3-4B-Base | random | 100 | 2000 | 0.607 | 0.210 | 0.247 | 0.297 | 0.276 | 0.602 | 0.265 |
| Qwen3-4B-Base | random | 150 | 2000 | 0.610 | 0.212 | 0.243 | 0.288 | 0.278 | 0.594 | 0.261 |
| Qwen3-4B-Base | random | 200 | 2000 | 0.602 | 0.211 | 0.243 | 0.290 | 0.276 | 0.598 | 0.266 |
| Qwen3-4B-Base | random | 300 | 2000 | 0.594 | 0.222 | 0.253 | 0.292 | 0.274 | 0.590 | 0.258 |
| Qwen3-8B† | random | 285 | 800 | 0.325 | 0.325 | 0.325 | 0.325 | 0.325 | 0.325 | 0.325 |
| Qwen3-8B-Base | random | 30 | 2000 | 0.615 | 0.306 | 0.306 | 0.317 | 0.320 | 0.594 | 0.306 |
| Qwen3-8B-Base | random | 45 | 2000 | 0.627 | 0.306 | 0.298 | 0.317 | 0.312 | 0.609 | 0.302 |
| Qwen3-8B-Base | random | 105 | 2000 | 0.618 | 0.326 | 0.316 | 0.335 | 0.336 | 0.598 | 0.303 |
| Qwen3-8B-Base | random | 150 | 2000 | 0.695 | 0.280 | 0.290 | 0.340 | 0.304 | 0.673 | 0.293 |
| Qwen3-8B-Base | random | 195 | 2000 | 0.669 | 0.273 | 0.279 | 0.348 | 0.300 | 0.638 | 0.303 |
| Qwen3-8B-Base | random | 300 | 2000 | 0.644 | 0.286 | 0.288 | 0.327 | 0.310 | 0.616 | 0.280 |