Time-for-Accuracy: Formalizing Chain-of-Thought as an Expansion of Logical Depth

Yue Wang[1*] Zhi Zhang[1*] Wang Xi[2*]
Chengjie Sun[1†] Lili Shan[1,3] Bingquan Liu[1]
[1]Faculty of Computing, Harbin Institute of Technology, Harbin, China
[2]Faculty of Computing, University of Science and Technology of China, Hefei, China
[3]National Research Center for Language Technology and Digital Economy,
Harbin Institute of Technology, Harbin, China
{wyue, zzhi}@stu.hit.edu.cn, xw_cs@mail.ustc.edu.cn
{sunchengjie, shanlili, liubq}@hit.edu.cn

Abstract

Chain-of-thought (CoT) often improves multi-step reasoning, but it remains unclear what kind of additional sequential computation longer traces actually enable. We connect CoT to Bennett's logical depth, separating an answer's description length from the sequential effort required to derive it, and view a CoT budget of T steps as a qualitative cap on realizable sequential computation. To operationalize realized depth beyond raw length, we introduce *Effective Logical Depth* (ELD), a deletion-based measure of step necessity under a specified inference interface. Across depth-controlled prefix-sum tasks and GSM8K rationale perturbations, we observe two consistent signatures of a *Time-for-Accuracy* tradeoff: (i) plateau-to-transition accuracy curves as the budget increases from being below to matching the task's required depth, and (ii) sparse, position-dependent deletion sensitivity concentrated in early steps for deeper instances. On GSM8K, an EXTRACT interface, where the model reads off the answer from the remaining rationale, remains near-perfect even after prefix deletions, whereas a REPAIR interface, where the model must re-solve from truncated rationale context, degrades markedly. Moreover, SOCRATIC human rationales are consistently more robust than MAIN rationales under REPAIR. These results suggest that longer CoT helps primarily when it enables additional effective sequential computation, and that deletion-based diagnostics can distinguish computational steps from redundant ones.

1 Introduction

Chain-of-thought (CoT) prompting can substantially improve multi-step reasoning in large language models (LLMs) (Wei et al., 2022; Kojima et al., 2022), and sampling/aggregation such as self-consistency can further increase reliability (Wang et al., 2022). These trends are often discussed as allocating more test-time compute (Wu et al., 2024; Snell et al., 2024; Bi et al., 2024), but gains may plateau without strong verification (Brown et al., 2024), and the mechanism of "longer traces ⇒ better reasoning" remains debated (Gan et al., 2025).

We focus on a basic ambiguity: an answer may be *informationally* simple (e.g., a short integer) but still *computationally* hard to derive from the input. Algorithmic information theory separates these notions: Kolmogorov complexity captures conditional description length, while Bennett's logical depth captures the runtime of near-shortest programs, i.e., required *sequential effort* (Li and Vitányi, 2008; Bennett, 1988). This lens suggests that CoT can be viewed as an externalized unrolling of sequential computation, and motivates asking: (i) how should a CoT budget relate to a task's intrinsic sequential depth, and (ii) within a chain, which steps are causally necessary vs. redundant?

We make three contributions:

- **Logical-depth framing.** We relate CoT budgets to *relative logical depth* (sequential effort) rather than description length.
- **Time-for-Accuracy prediction.** Under a budgeted protocol and a mild near-shortest-program bias, insufficient budget induces a depth-driven error plateau; once budget covers required depth, the forced computational floor disappears.
- **ELD + evidence.** We propose *Effective Logical Depth* (ELD), a deletion-based proxy for realized step necessity, and validate the predicted plateau-to-transition curves and sparse critical-step structure on depth-controlled prefix-sum tasks and GSM8K rationale deletions.

Beyond the conceptual framing, ELD also provides a lightweight, verifier-free diagnostic for rea-

*These authors contributed equally.
†Corresponding author: sunchengjie@hit.edu.cn

Findings of the Association for Computational Linguistics: ACL 2026, pages 38925–38941
July 2-7, 2026

soning traces. In settings where models may produce persuasive but potentially unfaithful rationales, deletion-based sensitivity can help distinguish traces that function as genuine intermediate computation from traces that merely provide redundant or answer-revealing context. This makes ELD potentially useful for faithfulness analysis, test-time compute allocation, and high-stakes auditing scenarios where external verification is costly. Additional appendix analyses further broaden the empirical picture, including evaluations on additional models and datasets, budgeted-continuation controls, format-preserving perturbations, and significance tests.

2 Framework: Relative Logical Depth and Effective Logical Depth

We introduce two notions that will be used throughout. First, we separate an answer's *information content* from the *sequential effort* needed to obtain it. We measure information content by conditional description length (Kolmogorov complexity), and sequential effort by *relative logical depth*. Second, we define a protocol-level, intervention-based proxy—*Effective Logical Depth* (ELD)—that quantifies how much a provided chain is *actually relied on* for correctness under a specified inference interface. ELD is tailored to the concrete evaluation protocols in §4, not to internal transformer mechanisms.

2.1 Budgeted protocols and relative logical depth

Protocol budget. Many CoT-style protocols impose an explicit budget on intermediate reasoning, e.g., "produce at most T steps and then answer". We treat T as a *protocol budget*, counted in the units enforced externally by the protocol (step lines, delimited segments, etc.). We do not equate T with the number of internal transformer operations (Merrill and Sabharwal, 2025). For our qualitative predictions we only need monotonicity: larger budgets permit longer externally realized sequences of state updates. For notational convenience, we write an abstract mapping $t = g(T)$ from protocol steps to an effective sequential-time scale, and we will use the linear form $g(T) = cT$ when stating informal statements. The constant $c > 0$ is not identified; setting $c = 1$ does not change any empirical claim.

Relative logical depth. Let x be an input, β be background knowledge (e.g., axioms or retrieved context), and y be an answer. Write $K(y \mid x, \beta)$ for the conditional Kolmogorov complexity of y given (x, β). Following Bennett, we define the *relative* (conditional) logical depth as the minimal runtime among *near-shortest* programs that derive y from (x, β):

$$\begin{aligned} LD_\delta(y \mid x, \beta) := \min_{t,p}\ & t \\ \text{s.t.}\quad & |p| \le K(y \mid x, \beta) + \delta, \\ & U^t(p, x, \beta) = y. \end{aligned} \tag{1}$$

Here U is a fixed universal machine, U^t denotes halting within t steps, and $\delta \ge 0$ is a significance parameter that restricts p to be within δ bits of the shortest conditional description, ruling out arbitrarily longer "hard-code y" descriptions. (See Appendix A for brief background on Kolmogorov complexity and logical depth.) This definition captures the regime that motivates CoT: an answer can be informationally short (small K) yet require many sequential updates to obtain from (x, β) (large LD_δ).

2.2 Interfaces, deletion sensitivity, and ELD

Traces and inference interfaces. A chain-of-thought trace is a sequence of externally delimited steps $r_{1:T} = (r_1, \ldots, r_T)$. How a trace influences the final answer depends on the inference procedure. We therefore model an *interface* $\mathcal{A}$ as the full protocol that maps (x, β, r) to an answer distribution (or a deterministic answer under fixed decoding). Concretely, $\mathcal{A}$ subsumes the prompt template, decoding rule, and any hard constraints (e.g., whether new intermediate steps are forbidden or allowed). This is the object that varies across our experiments (truncation with no new steps; delete-and-continue; Extract vs. Repair).

Deletion interventions and step sensitivity. Let $\mathrm{Del}_t(r_{1:T})$ remove step t from the trace (preserving order of the remaining steps). For an instance (x, β, y^*), let $y \sim \mathcal{A}(x, \beta, r_{1:T})$ and $y^{(-t)} \sim \mathcal{A}(x, \beta, \mathrm{Del}_t(r_{1:T}))$. We use a directional "break" indicator that counts correctness loss caused by deletion:

$$\mathrm{Break}(t) := \mathbf{1}\{y = y^* \ \wedge\ y^{(-t)} \neq y^*\}. \tag{2}$$

Define the step sensitivity (flip rate) as

$$\rho(t) := \mathbb{E}\big[\mathrm{Break}(t)\big], \tag{3}$$

where the expectation is over instances and any randomness induced by $\mathcal{A}$.

Two clarifications matter. (i) *Interface dependence is intended:* the same textual step can be essential under an interface that forces recomputation, yet nearly irrelevant under an interface that mainly reads off the answer from remaining context. (ii) Deletion is an input intervention and can introduce distribution shift or formatting artifacts. Accordingly, $\rho(t)$ should be interpreted as *protocol-level intervention sensitivity of correctness*, not as a ground-truth decomposition of "true reasoning."

Effective Logical Depth (ELD). We aggregate deletion sensitivities into a protocol-level realized-depth proxy:

$$\mathrm{ELD}_{\mathcal{A},T} := c \sum_{t=1}^{T} \rho(t). \tag{4}$$

Intuitively, $\mathrm{ELD}_{\mathcal{A},T}$ is the expected number (up to scale c) of step deletions that flip a correct prediction to incorrect under interface $\mathcal{A}$. When useful, one can also define the per-instance counterpart $\mathrm{ELD}(x, \beta, r_{1:T}) := c \sum_{t=1}^{T} \mathrm{Break}(t)$ and then recover (4) by taking expectations.

Why ELD is useful for our predictions. The theory in §3 is phrased in terms of an instance-level depth requirement $D(x, \beta, y^*) := LD_\delta(y^* \mid x, \beta)$, which is uncomputable in general. ELD is a behavioral proxy with narrower scope: it distinguishes long traces that function as computation from long traces that mostly provide redundant context. Our empirical link is qualitative rather than calibrated: below the required depth, accuracy exhibits a plateau as T increases; once the budget becomes sufficient, correctness becomes sensitive to a relatively small set of steps under deletion-based interventions. We test these signatures under three interfaces in §4 (Exp. 1–3).

3 Time-for-Accuracy Tradeoff

This section states a stylized but falsifiable link between a CoT *step budget* and achievable accuracy, which we refer to as a Time-for-Accuracy tradeoff. The core object is the instance-wise *required sequential depth* measured by relative logical depth. The resulting prediction is an accuracy *plateau* when the budget undershoots the depth mass of the task distribution, followed by a *transition* once the budget covers that mass.

Setup. Let $(x, \beta, y^*) \sim \mathcal{D}$ be an instance, where x is the input, β is background knowledge, and y^* is the (unique) correct answer. Fix a universal machine U and a significance parameter δ. Define the required depth as

$$D(x, \beta, y^*) \;:=\; LD_\delta(y^* \mid x, \beta). \tag{5}$$

Consider a protocol/interface that limits the model to at most T externally delimited reasoning steps before producing the final answer. Under the budget abstraction in §2.1, we associate this with an effective sequential-time budget $t = cT$ for some constant $c > 0$. Let h_T^θ denote a (possibly randomized) model instantiated under this T-step protocol. We measure final-answer error by

$$\mathrm{Err}_T(h_T^\theta; \mathcal{D}) := \mathbb{E}_{(x,\beta,y^*)\sim\mathcal{D}} \; \mathbb{E}\Big[\mathbf{1}\{h_T^\theta(x, \beta) \neq y^*\}\Big]. \tag{6}$$

where the inner expectation is over any randomness in decoding/protocol. Define the best achievable error at budget T within the model class as

$$\mathrm{Err}_T^* := \inf_\theta \; \mathrm{Err}_T(h_T^\theta; \mathcal{D}). \tag{7}$$

Depth mass under the task distribution. A convenient way to summarize how "deep" the task is at budget t is the *deep-instance mass*

$$\epsilon(t) \;:=\; \Pr_{(x,\beta,y^*)\sim\mathcal{D}}\big[\, D(x, \beta, y^*) > t \,\big]. \tag{8}$$

When $\epsilon(t)$ stays large over a range of budgets, improvements from increasing T can be limited for purely computational reasons.

A qualitative bias assumption. The definition of D allows arbitrarily long, instance-specific "hard-code" strategies to bypass depth. To connect D to learned models, we assume the inference strategy is not dominated by such lookup behavior.

Assumption 1 (Near-shortest-program bias (qualitative))**.** Under a T-step protocol (effective budget $t = cT$), the model's behavior is governed by a reusable procedure whose description length is not inflated to encode a large fraction of instance-specific answers. Equivalently, among strategies consistent with training, the learned solution behaves like selecting from *near-shortest* procedures that must halt within the available budget t.

Proposition 1 (Time-for-Accuracy tradeoff (stylized, informal))**.** *Fix δ and consider a T-step protocol with effective budget $t = cT$. Under Assumption 1:*

1. ***Undershooting depth induces a non-zero floor.*** *The best achievable error under the protocol is lower-bounded (up to modeling details) by the deep-instance mass:*

$$\mathrm{Err}_T^* \gtrsim \epsilon(cT). \tag{9}$$

In particular, if $\epsilon(cT) \geq \epsilon_0 > 0$ over a range of T, then a non-zero depth-induced *error plateau is unavoidable in that range.*

2. ***Once depth is covered, there is no forced computational floor.*** *If $\epsilon(cT) = 0$ (i.e., $D(x, \beta, y^*) \leq cT$ holds $\mathcal{D}$-a.s.), then the definition of D implies that near-shortest correct procedures exist within the available budget for all instances. Hence any remaining error floor is not compelled by sequential-depth constraints; residual errors are attributable to statistical/optimization effects, distribution shift, or protocol artifacts rather than a lack of sequential budget.*

Empirical signatures and the role of ELD. Proposition 1 motivates two signatures that we test: **(P1)** plateau-to-transition accuracy curves as T increases, consistent with the shape of $\epsilon(cT)$; and **(P2)** sparse critical-step structure under deletion interventions once the plateau is escaped.

To connect these signatures to observed traces, we use the protocol-level realized-depth proxy $\mathrm{ELD}_{\mathcal{A},T} = c\sum_{t=1}^{T} \rho(t)$ from Eq. 4, defined for a fixed interface $\mathcal{A}$. Intuitively, increasing T only helps insofar as it increases *realized* sequential dependence (ELD), not merely raw trace length. Accordingly, near the transition region we expect (i) substantial increases in $\mathrm{ELD}_{\mathcal{A},T}$ with T, and (ii) deletion sensitivity concentrated on a relatively small subset of positions. We evaluate these signatures in §4 (Exp. 1–3).

4 Experiments

We test two qualitative signatures *motivated by* §3: (P1) plateau-to-transition Time-for-Accuracy curves as the CoT budget T increases, and (P2) sparse, *position-dependent* deletion sensitivity that strengthens with depth (ELD), noting that deletion effects are intentionally *interface-dependent* (§2.2). Exp. 1 evaluates (P1) on a depth-controlled prefix-sum family; Exp. 2 estimates deletion sensitivity via delete-and-continue on the same family; Exp. 3 applies analogous perturbations to GSM8K human rationales under different interfaces. To broaden scope and address robustness concerns, the appendix further reports extended validations on additional models and datasets, budgeted-continuation controls, format-preserving perturbations, and statistical significance analyses.

4.1 Depth-controlled prefix-sum tasks

Task and modes. Given $(a_1, \ldots, a_n)$ with $a_i \sim \mathrm{Unif}([-9, 9])$, the target is $b_n = \sum_{i=1}^{n} a_i$ (a short integer), while a natural derivation requires n sequential updates. We use $n \in \{4, 8, 16, 32\}$ with 500 instances per depth and exact-match evaluation on `Answer: <integer>`.

We compare **No-CoT** (direct answer), **Full-CoT** (emit n step lines then answer), and **Trunc-CoT**: the model is shown only the first T step lines from a corresponding Full-CoT trace ($T < n$) *and is forbidden to generate any new intermediate steps*, forcing the remaining $(n - T)$ updates to be compressed into a single-shot final answer. Unless stated otherwise, Trunc-T is evaluated on the Full-correct & format-valid subset to reduce confounds and isolate depth-budget effects under this "no-new-steps" constraint.

4.2 Experiment 1: Depth-driven Time-for-Accuracy Curves

Goal. We test the central qualitative prediction of Time-for-Accuracy (§3) on a depth-controlled family where the required *sequential* computation scales with task depth, while the final answer remains a short integer. We use the prefix-sum tasks from §4.1: given $(a_1, \ldots, a_n)$ with $a_i \sim \mathrm{Unif}([-9, 9])$, the model must output the final prefix sum $b_n = \sum_{i=1}^{n} a_i$.

Data and model. For each depth $n \in \{4, 8, 16, 32\}$ we sample $N = 500$ instances. Unless otherwise noted, we use Llama-3.1-8B-Instruct with greedy decoding (temperature $\tau = 0$). We additionally replicate Experiment 1 on Qwen2.5-7B-Instruct under the same protocols and observe the same qualitative pattern; see Appendix B.

Prompting protocols. We compare three regimes: **No-CoT** (T=0) outputs only `Answer: <integer>`. **Full-CoT** (T=n) outputs step lines `Step` k: $b_k = \cdots$ for $k = 1 \ldots n$ and then `Answer:` b_n. **Trunc-CoT** (Scheme A) provides the model with the first T step lines *taken from a Full-CoT output on the same instance*, and requires outputting the final answer *without generating any new intermediate steps*.

Answer parsing and "strict" end-to-end evaluation. Final answers are evaluated using *robust answer parsing*: we preferentially read an `Answer:` line if present (taking the last integer after the tag), and otherwise fall back to the last integer in the output (Appendix B). For Full-CoT, we additionally track **step-pass**: whether the output contains a parsable and complete step trace (covering `Step 1` through `Step` n under our parser). We report: (i) *Full (valid)* accuracy on the step-pass subset; and (ii) *Full (strict)* accuracy where step-fail outputs are counted as incorrect. We also report the *Full step-pass rate* to separate formatting brittleness from arithmetic correctness.

Trunc-CoT evaluation subset. To reduce confounds from trace-generation failures, Trunc-T is evaluated only on instances where the corresponding Full-CoT run is *(a) step-pass and (b) answer-correct*. Let M_n denote this subset size at depth n; in our Llama runs, $M_4 = 500$, $M_8 = 429$, $M_{16} = 375$, and $M_{32} = 288$ (Table 1, Appendix B). We further audit intermediate-state consistency of these prefixes in Appendix B.

Results vs. depth. Figure 1 plots accuracy as a function of depth n. No-CoT collapses as depth increases ($0.344 \rightarrow 0.012$ from n=4 to n=32), consistent with a depth-limited regime where a single short answer emission cannot reliably implement long sequential updates. In contrast, Full-CoT remains strong when it step-passes: Full(valid) stays high across depths ($1.000, 0.858, 0.880, 0.785$ for $n = \{4, 8, 16, 32\}$). However, Full(strict) drops more at larger depths (e.g., 0.750 at n=16 and 0.576 at n=32) due to decreasing step-pass rates ($1.000, 1.000, 0.852, 0.734$).

Results vs. budget: plateau $\rightarrow$ jump. Figure 2 fixes n and varies the available budget T. For deeper instances (n=16, 32), accuracy stays near a *depth-induced plateau* for small budgets (No-CoT and Trunc-CoT at small T are near 0), and exhibits a sharp *transition* when the budget reaches the task depth (T=n), matching the qualitative Time-for-Accuracy prediction: below the required sequential depth, additional context that does not permit further unrolling yields limited gains; once the budget suffices to realize the full update sequence, accuracy increases abruptly.

Why is Trunc-CoT low? Trunc-CoT is intentionally stringent: conditioned on a truncated step prefix from a Full-CoT output, the model is forbidden from emitting additional intermediate steps and must compress the remaining $(n - T)$ updates into a single answer emission. Empirically this yields very low accuracy on deeper settings under Scheme A (e.g., at n=32, Trunc-2 is 0.003 and Trunc-4 is 0.000 on the same $M_{32} = 288$ instances). Appendix B reports sanity checks showing that (i) the provided b_T in Trunc inputs matches the oracle prefix sum, and (ii) on the evaluated Trunc subset, step-wise inconsistencies inside the Full-CoT prefixes are rare at $n \in \{16, 32\}$, supporting the interpretation that Trunc failures primarily reflect insufficient effective depth budget under the "no-new-steps" constraint.

n	No-CoT	Tr-2†	Tr-4†	Full (strict / valid)
4	0.344	0.458	–	1.000 / 1.000
8	0.082	0.086	0.175	0.858 / 0.858
16	0.038	0.029	0.013	0.750 / 0.880
32	0.012	0.003	0.000	0.576 / 0.785

Table 1: Experiment 1 on prefix-sum (Llama-3.1-8B-Instruct). Full reports strict end-to-end accuracy (step-fail counted as wrong) and valid-only accuracy (step-pass subset). Full step-pass rates for $n \in \{4, 8, 16, 32\}$ are 1.000, 1.000, 0.852, 0.734. Trunc-$T^\dagger$ accuracies are computed on the subset M_n where the corresponding Full-CoT run is step-pass and answer-correct, so denominators differ across methods and depths.

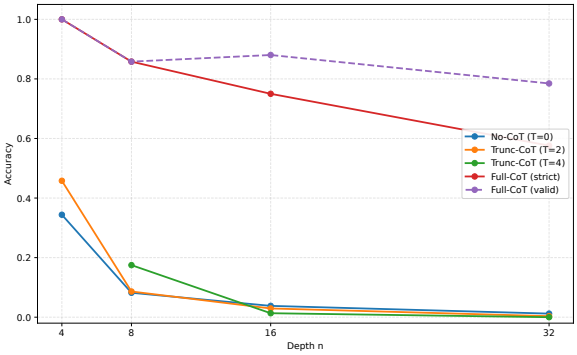

Figure 1: Accuracy vs. task depth n under No-CoT (T=0), Trunc-CoT ($T \in \{2, 4\}$), and Full-CoT (T=n). Full(strict) counts step-fail outputs as incorrect; Full(valid) is computed on the step-pass subset. Trunc-T is evaluated on the subset M_n where the corresponding Full-CoT run is step-pass and answer-correct, so denominators differ across methods and depths. (Llama-3.1-8B-Instruct, greedy decoding; Qwen2.5-7B replication in Appendix B.)

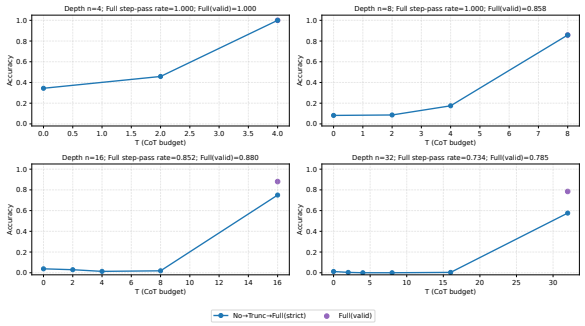

Figure 2: Accuracy vs. CoT budget T at fixed depths $n \in \{4, 8, 16, 32\}$. Here T=0 corresponds to No-CoT; $0 < T < n$ to Trunc-T under Scheme A (no-new-steps); and T=n to Full-CoT. The curve reports strict end-to-end accuracy for each regime; the marker at T=n additionally indicates Full(valid-only) accuracy on the step-pass subset. Trunc points are evaluated on M_n (Full step-pass & answer-correct), so denominators are not identical across all T. (Qwen replication: Appendix B.)

4.3 Experiment 2: Deletion Tests on Self-Generated Chains-of-Thought

Motivation. Experiment 4.2 establishes depth-driven time-for-accuracy curves on a depth-controlled family. Here we probe *where* successful computation becomes fragile by deleting an intermediate step of a correct Full-CoT trace and asking the model to continue to the final answer. If a chain functions as a true computational trace, removing early steps should disproportionately disrupt later reasoning, especially at larger depths.

Models and subset construction (from Exp. 1). We evaluate two instruction-tuned LLMs: Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct. For each depth $n \in \{4, 8, 16, 32\}$ and each model, we start from Experiment 1 Full-CoT outputs and keep only the **strict-correct** subset: (i) the trace satisfies the required `Step` k`:` formatting through `Step` n, and (ii) the implied final answer is correct. From this strict-correct pool we then **uniformly sample 150 instances per depth** (fixed seed), yielding at most $150 \times n$ deletion trials per depth before filtering.

Delete-and-Continue interface (Option B; code-faithful). Let $(a_1, \ldots, a_n)$ be the input integers and define prefix sums by $b_0 = 0$ and $b_i = b_{i-1} + a_i$. For each sampled instance with a strict-correct Full-CoT trace containing step lines `Step` i`:` $\cdots b_i \cdots$, we construct a deletion variant indexed by a deleted position $t \in \{1, \ldots, n\}$: we *hide* step t and all later steps and show only steps $1, \ldots, t-1$. We also provide the full input list $a_{1:n}$ and reprint the tail addends $a_{t:n}$ to align the continuation prompt (this reveals no information beyond $a_{1:n}$). The model is asked to continue the computation from the last visible state and output the final value b_n.

Unlike Scheme A in Exp. 1 (which forbids intermediate steps), we *allow* the model to produce reasoning text for analysis, but require an explicit final line `Answer: <integer>`. If the first attempt does not contain a parseable `Answer:` line, we re-ask *with the same user content* but a stricter system instruction that forces a single-line answer (one retry). All runs use greedy decoding ($\tau = 0$).

Answer parsing and bad events. We parse predictions *only* from the integer appearing after an `Answer:` tag (no fallback to the last integer elsewhere). A deletion trial is labeled **bad** if the parsed answer is incorrect. If no parseable `Answer:` line is produced, we treat it as **bad** for ρ_{all} and as **invalid** for ρ_{valid}.

Under our subset construction (conditioning on Full-CoT strict-correct baselines), a bad deletion trial corresponds to a break event in Eq. 2; thus the reported bad rates are filtered estimates of $\rho(t)$ in Eq. 3 under this delete-and-continue interface.

Shortcut-free filtering. Some deletions are degenerate because copying the last visible prefix sum already equals the correct final answer. We therefore exclude such trials and report *shortcut-free* results by filtering out deletions where

$$b_{t-1} = b_n \quad \Longleftrightarrow \quad \sum_{i=t}^{n} a_i = 0. \tag{10}$$

Let $N_{\text{sf,valid}}$ denote the number of remaining *deletion trials* (instance $\times$ deleted position) that are both shortcut-free and valid-only.

Region aggregation and copy decomposition. To summarize deletion sensitivity compactly, we bin deleted positions by normalized location t/n: Early $[0, 0.15]$, Mid $(0.15, 0.70]$, and Late $(0.70, 1]$. For each region we report the shortcut-free, valid-only bad rate ρ_{sf}^{Region} aggregated over all trials whose t falls in that bin. We also track a structured failure mode, *nontrivial copy*: the model outputs the last visible state b_{t-1} as the final answer even when the shortcut-free condition holds (i.e., $\sum_{i=t}^{n} a_i \neq 0$). In the Early region we report the absolute rate of this failure mode, denoted Copy^{Early};

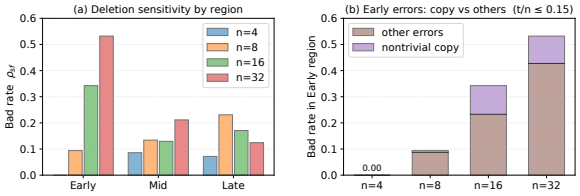

Figure 3: **Deletion sensitivity by region (Llama-3.1-8B-Instruct, delete-and-continue / Option B).** (a) Shortcut-free, valid-only bad rate ρ_{sf} aggregated by normalized deletion position t/n into Early $[0, 0.15]$, Mid $(0.15, 0.70]$, and Late $(0.70, 1]$ for depths $n \in \{4, 8, 16, 32\}$. (b) Early-region bad rate decomposed into *nontrivial copy* (predicting the last visible state b_{t-1} as the final answer despite $\sum_{i=t}^{n} a_i \neq 0$) versus other errors; stacked bars sum to ρ_{sf}^{Early}. The Early bin is empty for n=4 since $t/n \geq 0.25$ for all $t \in \{1, \ldots, 4\}$. Rates are computed over deletion trials (instance $\times$ deleted position) after shortcut-free filtering and discarding invalid outputs without a parseable `Answer:` line.

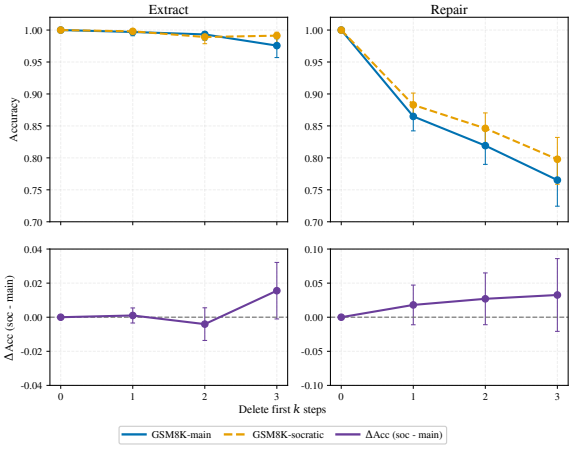

Figure 4: Experiment 3 prefix truncation robustness on GSM8K (delete first k steps). Points are computed on policy-specific used subsets (varying denominators; Appendix D). We also report ΔAcc (SOCRATIC−MAIN) for each policy.

the ratio $\text{Copy}^{Early}/\rho_{sf}^{Early}$ is the fraction of Early-region errors attributable to copying.

Results. Figure 3 shows that deletion sensitivity concentrates in the *Early* region and strengthens with depth. For Llama, ρ_{sf}^{Early} increases from 0.094 at n=8 to 0.532 at n=32, while Mid/Late remain substantially lower (Table 2). Panel (b) indicates that nontrivial copy explains only part of early failures (e.g., $\text{Copy}^{Early}/\rho_{sf}^{Early} \approx 32\%$ at n=16 and $\approx 20\%$ at n=32), with the remainder attributable to other continuation errors under deletion. We observe the same qualitative trend on Qwen; see Appendix C.

4.4 Experiment 3: GSM8K Deletion Tests on Human Chains-of-Thought

Goal. We test whether deletion-based signatures transfer from synthetic prefix-sum traces to natural-language rationales. We use GSM8K test problems with human-written rationales in two styles (MAIN, SOCRATIC) and study how deleting early steps affects answer correctness under different inference interfaces.

Setup and interfaces. We evaluate 1,000 GSM8K test instances per rationale style using the same model and greedy decoding as Exp. 1–2. Rationales are segmented into discrete steps by a sentence/line splitter (Appendix D). We compare: **No-CoT** (question only), **Extract** (rationale only; question removed), and **Repair** (question + rationale). For prefix truncation we delete the first $k \in \{1, 2, 3\}$ steps.

Well-posedness and conditioning. To prevent EXTRACT from degenerating into re-solving, we enforce a support constraint: the predicted number must appear in the *kept* rationale steps; instances violating this after deletion are excluded for that condition. To isolate deletion effects rather than baseline failures, we report results on policy-specific *used* subsets that condition on being correct at k=0 under the same policy (so the k=0 rows are 1.0 by construction; unconditioned baselines are in Appendix D).

Results. Fig. 4 shows a sharp separation between EXTRACT and REPAIR. Under EXTRACT, accuracy stays high after prefix truncation (e.g., $\geq$ 0.976 up to k=3), consistent with the model mainly extracting the final answer from suffix cues in the remaining rationale. Under REPAIR, performance degrades substantially as early steps are removed (e.g., from 1.0 at k=0 to 0.765/ 0.798 at k=3 for MAIN/SOCRATIC), indicating that truncated rationales provide incomplete intermediate context that can hinder re-solving. Across k, SOCRATIC is consistently more robust under REPAIR (about +2–3 points). Additional window-deletion flip-rate analyses are reported in Appendix D.6.

4.5 Discussion

Across the three settings, we see a consistent picture: longer chains help only when they support

Model	n	$N_{\text{sf,valid}}$	Early ρ_{sf}	Mid ρ_{sf}	Late ρ_{sf}	All ρ_{sf}	Copy$^{\text{Early}}$
Llama-3.1-8B-Instruct	4	570	–	0.086	0.072	0.079	–
	8	1160	0.094	0.134	0.231	0.165	0.007
	16	2327	0.342	0.129	0.171	0.169	0.110
	32	4706	0.532	0.211	0.124	0.225	0.105
Qwen2.5-7B-Instruct	4	570	–	0.010	0.014	0.012	–
	8	1155	0.014	0.017	0.021	0.018	0.000
	16	2318	0.239	0.099	0.090	0.113	0.003
	32	4694	0.737	0.256	0.148	0.284	0.069

Table 2: Experiment 2 summary (shortcut-free, valid-only) for Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct. For each depth n and model, we sample 150 Full-CoT *strict-correct* instances from Exp. 1, yielding at most $150 \times n$ deletion trials before filtering. $N_{\text{sf,valid}}$ is the number of remaining trials (instance $\times$ deleted position) after (i) shortcut-free filtering ($b_{t-1} \neq b_n$) and (ii) discarding outputs without a parseable `Answer:` line. Regions use normalized position t/n: Early $[0, 0.15]$, Mid $(0.15, 0.70]$, Late $(0.70, 1]$. '–' indicates an empty region (for n=4, Early is empty since $t/n \geq 0.25$ for all $t \in \{1, \ldots, 4\}$). Copy$^{\text{Early}}$ is the absolute rate of *nontrivial copy* within Early deletions (predicting b_{t-1} as the final answer while $\sum_{i=t}^{n} a_i \neq 0$).

additional *effective sequential computation*, rather than merely adding context. In Exp. 1, the depth-controlled prefix-sum family shows clear plateau-to-transition behavior: truncated prefixes still contain correct intermediate states, but forbidding any further steps largely restores a depth-induced error floor, with accuracy jumping only when the budget reaches the full depth.

Exp. 2 then localizes where this computation is fragile: as depth increases, deleting *early* steps is much more disruptive than deleting later ones, and a noticeable fraction of errors follow shallow shortcuts such as copying the last visible state, showing that long traces do not automatically imply deep computation. Exp. 3 applies similar perturbations to GSM8K human rationales and separates two ways chains are used: under EXTRACT, the model mainly behaves as an answer extractor with robustness to prefix truncation, whereas under REPAIR, removing early prefixes causes substantial accuracy drops, with SOCRATIC consistently more robust than MAIN (Appendix D.6).

Overall, these results suggest that token-based "slow thinking" is beneficial insofar as it increases realized sequential depth, and that deletion-style interventions (ELD) offer a lightweight way to distinguish computation from redundancy under a given interface. See the Limitations section for caveats on protocol dependence and scope. The broader appendix results are consistent with this interpretation: the same qualitative separation between answer extraction and re-computation persists on additional benchmarks, while control experiments show that the observed degradation cannot be reduced to formatting shift alone.

5 Related Work

CoT and test-time compute. Chain-of-thought prompting improves multi-step reasoning by externalizing intermediate traces (Wei et al., 2022; Kojima et al., 2022), with sampling-based aggregation such as self-consistency further increasing robustness (Wang et al., 2022). Recent work frames these gains as *test-time compute scaling*, where allocating additional inference compute (tokens, samples, or search) can improve accuracy under fixed model size (Wu et al., 2024; Snell et al., 2024; Zhang et al., 2025a), though selection without strong verification can plateau (Brown et al., 2024; Wang et al., 2024a). Our work complements this line by focusing on a single-chain axis: how a bounded chain budget T relates to an instance's intrinsic *sequential depth* requirement.

Step-level diagnostics of reasoning traces. A growing literature studies intermediate steps via step-level evaluation and trajectory assessment (Zheng et al., 2025), and via learned process-level signals such as process reward models (PRMs) that can score steps/trajectories and support selection (Khalifa et al., 2025; Wang et al., 2025; Zhang et al., 2025b; Chen et al., 2024; Xi et al., 2025). In contrast, we introduce a verifier-free, *deletion-based* diagnostic (ELD) that estimates which steps are *causally necessary* for correctness under a specified interface, linking raw chain length to *realized* sequential computation.

Faithfulness and theoretical perspective. Concerns about when long rationales reflect genuine computation motivate robustness and faithfulness analyses of CoT (Chen et al., 2025; Prabhakar et al.,

2024; Stechly et al., 2024; Balasubramanian et al., 2025). We connect these empirical questions to algorithmic information theory: while Kolmogorov complexity captures description length, Bennett's logical depth captures sequential effort via near-shortest programs (Li and Vitányi, 2008; Bennett, 1988), enabling our Time-for-Accuracy view and its deletion-based operationalization.

6 Conclusion

We presented a logical-depth perspective on chain-of-thought (CoT) that separates an answer's description length from the sequential effort required to derive it. Under a depth-budget abstraction, this leads to a Time-for-Accuracy view: insufficient chain budget induces a depth-driven error plateau, while matching the required sequential depth removes this forced computational floor. We instantiated realized sequential depth with a deletion-based proxy, Effective Logical Depth (ELD), and observed the predicted plateau-to-transition curves and sparse, position-dependent step sensitivity on depth-controlled prefix-sum tasks and GSM8K rationale perturbations. Overall, our results suggest that longer traces are most useful when they increase *effective* sequential computation, and that deletion-style interventions offer a simple diagnostic for distinguishing computation from redundancy under concrete inference interfaces.

Limitations

Depth budget is qualitative. Our mapping from CoT step length to effective sequential computation is intentionally abstract and not quantitatively calibrated across models, decoding strategies, or step granularities. The Time-for-Accuracy view and the associated proposition are therefore best read as an explanatory toy model rather than a mechanistic account of transformer internals.

ELD is protocol-dependent and conflates factors. Deletion-based necessity is defined relative to the inference interface (e.g., EXTRACT vs. REPAIR) and can reflect distribution shift, state-tracking brittleness, or reliance on surface cues in addition to genuine computational dependence. ELD should thus be interpreted as a behavioral causal-sensitivity measure under a given protocol, not as a ground-truth decomposition of "true reasoning steps."

Scope and evaluation conditioning. Our strongest evidence comes from synthetic arithmetic and GSM8K grade-school math, and does not cover domains such as code generation, long-horizon planning, tool use, or open-domain multi-hop QA. Several analyses condition on well-formed and correct full traces to isolate depth effects, which may bias toward easier instances and interacts with formatting robustness.

Ethical Considerations

Our study uses synthetic arithmetic data and publicly available GSM8K problems and human-written rationales, and does not involve personal or sensitive information.

Risk of over-trust in rationales. Our results highlight that the same chain can support different behaviors (e.g., extracting cues versus recomputing), so long CoT explanations should not automatically be treated as faithful evidence of reasoning, especially in high-stakes settings. When model decisions matter, independent verification or external checking remains important.

Potential misuse and compute cost. Deletion-based diagnostics could in principle be used to craft persuasive yet unfaithful rationales or to probe model vulnerabilities. In addition, perturbation-style evaluation requires extra inference compute. Practical deployments should apply such tests selectively (e.g., on sampled or flagged instances) and, where possible, pair them with verification or human oversight.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 72532001.

References

Sriram Balasubramanian, Samyadeep Basu, and Soheil Feizi. 2025. A closer look at bias and chain-of-thought faithfulness of large (vision) language models. *arXiv preprint arXiv:2505.23945*.

Kenza Benkirane, Laura Gongas, Shahar Pelles, Naomi Fuchs, Joshua Darmon, Pontus Stenetorp, David Ifeoluwa Adelani, and Eduardo Sánchez. 2024. Machine translation hallucination detection for low and high resource languages using large language models. *arXiv preprint arXiv:2407.16470*.

Charles H. Bennett. 1988. Logical depth and physical complexity.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2024. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *arXiv preprint arXiv:2412.09078*.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. Alphamath almost zero: process supervision without process. *Advances in Neural Information Processing Systems*, 37:27689–27724.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

Zeyu Gan, Yun Liao, and Yong Liu. 2025. Rethinking external slow-thinking: From snowball errors to probability of correct reasoning. *arXiv preprint arXiv:2501.15602*.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2025. Process reward models that think.

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

Ming Li and Paul Vitányi. 2008. *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer.

Jianqiao Lu, Zhiyang Dou, Hongru Wang, Zeyu Cao, Jianbo Dai, Yunlong Feng, and Zhijiang Guo. 2024. Autopsv: Automated process-supervised verifier. *Advances in Neural Information Processing Systems*, 37:79935–79962.

William Merrill and Ashish Sabharwal. 2025. A little depth goes a long way: The expressive power of log-depth transformers. *arXiv preprint arXiv:2503.03961*.

Akshara Prabhakar, Thomas L Griffiths, and R Thomas McCoy. 2024. Deciphering the factors influencing the efficacy of chain-of-thought: Probability, memorization, and noisy reasoning. *arXiv preprint arXiv:2407.01687*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024. Chain of thoughtlessness? an analysis of cot in planning. *Advances in Neural Information Processing Systems*, 37:29106–29141.

Weiyun Wang, Zhangwei Gao, Lianjie Chen, and Zhe Chen. 2025. Visualprm: An effective process reward model for multimodal reasoning. *arXiv preprint arXiv:2503.10291*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yiming Wang, Pei Zhang, Baosong Yang, Derek F Wong, and Rui Wang. 2024a. Latent space chain-of-embedding enables output-free llm self-evaluation. *arXiv preprint arXiv:2410.13640*.

Zihan Wang, Yunxuan Li, Yuexin Wu, Liangchen Luo, Le Hou, Hongkun Yu, and Jingbo Shang. 2024b. Multi-step problem solving through a verifier: An empirical analysis on model-induced process supervision. *arXiv preprint arXiv:2402.02658*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Yajie Yang, Wenxiang Chen, Zhihao Zhang, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, and 1 others. 2025. Agentprm: Process reward models for llm agents via step-wise promise and progress. *arXiv preprint arXiv:2511.08325*.

Ori Yoran, Kunhao Zheng, Fabian Gloeckle, Jonas Gehring, Gabriel Synnaeve, and Taco Cohen. 2025. The kolmogorov test: Compression by code generation. *arXiv preprint arXiv:2503.13992*.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, and 1 others. 2025a. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, and 1 others. 2025b. Process vs. outcome reward: Which is better for agentic rag reinforcement learning. *arXiv preprint arXiv:2505.14069*.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1009–1024.

A Additional preliminaries and intuition

This appendix provides brief background on Kolmogorov complexity and Bennett's logical depth, supporting the definitions in §2 (see also Li and Vitányi, 2008; Bennett, 1988).

A.1 Kolmogorov complexity (definitions)

Let U be a fixed universal Turing machine. The (plain) Kolmogorov complexity of a string x is

$$K_U(x) := \min_p \{\, |p| \mid U(p) = x \,\}. \quad (11)$$

The conditional complexity of x given auxiliary information y is

$$K_U(x \mid y) := \min_p \{\, |p| \mid U(p, y) = x \,\}. \quad (12)$$

We drop the subscript U when the reference machine is clear from context.

A.2 Logical depth (absolute) and canonical regimes

(Absolute) logical depth. Fix a significance parameter $\delta \geq 0$. The δ-logical depth of a string x is

$$LD_\delta(x) := \min_{t,p} \; t \quad \text{s.t.} \quad |p| \leq K(x) + \delta, \quad U^t(p) = x. \quad (13)$$

The parameter δ restricts p to be within δ bits of the shortest description, ruling out arbitrarily longer "hard-code x" programs (which would make any string appear shallow).

Canonical regimes. Logical depth usefully distinguishes:

- **Simple and shallow:** low $K(x)$ and low $LD_\delta(x)$.
- **Random and shallow:** typically $K(x) \approx |x|$ but small depth (a near-shortest program can print x).
- **Structured and deep:** relatively low $K(x)$ yet large $LD_\delta(x)$ due to long unfolding computation.

A.3 Connection to the conditional notion used in this paper

Our paper uses a *conditional* (relative) notion of depth from an input (x, β) to an answer y. Rather than repeating the definition, we refer to Eq. 1 in §2. Conceptually, it replaces $K(\cdot)$ with $K(\cdot \mid x, \beta)$ and measures the minimal runtime among δ-near-shortest programs that map (x, β) to y.

B Sanity checks and cross-model replication for Experiment 1

Robust answer parsing (evaluation note). We evaluate final answers using robust parsing: we preferentially read an `Answer:` line if present (taking the last integer after the tag), and otherwise fall back to the last integer in the output. We apply this consistently to all regimes in Experiment 1.

Full-CoT intermediate-state consistency. We parse each `Step` k line to obtain $\hat{b}_k$ and compare it to the oracle prefix sum $\sum_{i=1}^{k} a_i$. Across all Full-CoT outputs, *any-step* mismatches are not rare (mismatch rate: 0.4% at n=4, 19.4% at n=8, 23.8% at n=16, and 39.8% at n=32 when counting whether an example contains at least one mismatching step). Crucially, on the Full-CoT subset used to evaluate Trunc-CoT (answer-correct; sizes M_n), mismatch rates drop sharply: 0.4% (2/500) at n=4, 6.3% (27/429) at n=8, 0.8% (3/375) at n=16, and 0.35% (1/288) at n=32. Thus, Trunc-CoT is typically conditioned on oracle-consistent intermediate states in the evaluated subset.

Trunc input sanity and low-regime non-monotonicity. We audit the provided b_T values in Trunc inputs against oracle prefix sums, and we check monotonicity on the *common-id intersection* across budgets. While Trunc accuracies at larger depths are near zero and are not strictly monotonic (e.g., at n=16 on the same 375 examples, Trunc-2 is 0.029 with Wilson 95% CI $[0.016, 0.052]$ whereas Trunc-4 is 0.013 with CI $[0.006, 0.031]$), these differences occur in the extreme low-accuracy regime and are quantified with uncertainty intervals.

Cross-model replication on Qwen2.5-7B-Instruct. To verify that the depth-driven Time-for-Accuracy pattern is not specific to a single instruction-tuned model, we replicate Experiment 1 with Qwen2.5-7B-Instruct under the same data generation and prompting protocols. We observe the same qualitative phenomenon: No-CoT and Trunc-CoT collapse rapidly with depth, whereas Full-CoT remains strong when it step-passes, with an increasing strict-vs-valid gap at larger depths driven by reduced step-pass rates. Concretely, Qwen Full-CoT valid-only accuracies are $1.000, 0.998, 0.976, 0.697$ for $n = \{4, 8, 16, 32\}$, while Full step-pass rates are $1.000, 1.000, 0.986, 0.714$.

n	#Step-pass	Step-pass rate	Full(valid)	Full(strict)
4	500	1.000	1.000	1.000
8	500	1.000	0.998	0.998
16	493	0.986	0.976	0.962
32	357	0.714	0.697	0.498

Table 3: Qwen2.5-7B-Instruct replication on Exp1 (prefix-sum). Step-pass rate is the fraction of outputs satisfying the required `Step` k: formatting checks. Full(valid) is accuracy on the step-pass subset; Full(strict) counts step-fail outputs as incorrect.

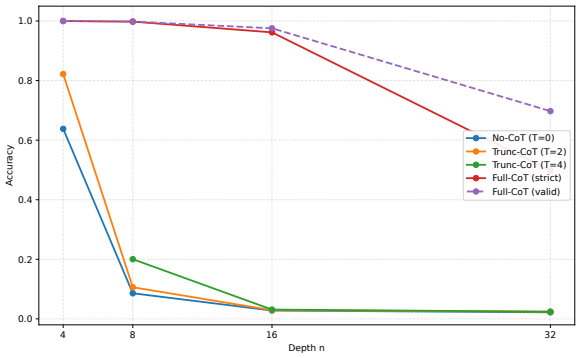

Figure 5: Experiment 1 replication (Qwen2.5-7B-Instruct): accuracy vs. depth n under No-CoT, Trunc-CoT, and Full-CoT. Full(strict) counts step-fail outputs as incorrect; Full(valid) is computed on the Full step-pass subset.

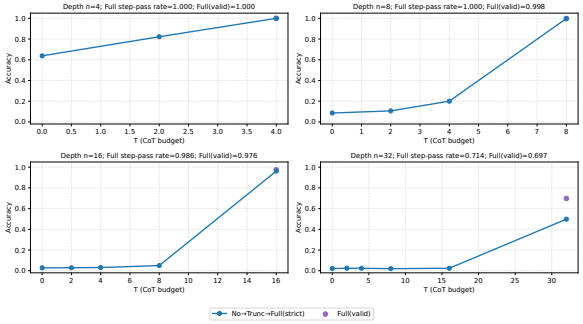

Figure 6: Experiment 1 replication (Qwen2.5-7B-Instruct): accuracy vs. CoT budget T at fixed depths. Titles report Full *step-pass* rate and Full(valid-only) accuracy.

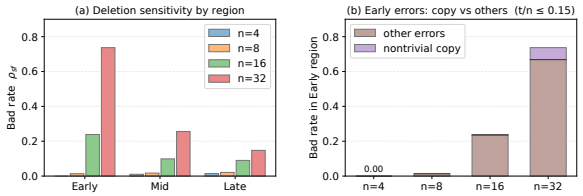

Figure 7: **Experiment 2 on Qwen2.5-7B-Instruct (same delete-and-continue / Option B protocol as Fig. 3).** (a) Shortcut-free, valid-only bad rate ρ_{sf} aggregated by deletion region (Early/Mid/Late defined by t/n). (b) Early-region errors decomposed into nontrivial copy ($\hat{y} = b_{t-1}$ while $\sum_{i=t}^{n} a_i \neq 0$) versus other errors. As in Fig. 3, the Early bin is empty for n=4.

C Additional Results for Experiment 2

Discussion. Qwen exhibits the same qualitative pattern: deletion sensitivity concentrates in the Early region and amplifies with depth (Table 2). Compared to Llama, Qwen shows a more extreme early-deletion sensitivity at n=32 (0.737), while the nontrivial-copy component remains relatively small, suggesting that most early failures arise from non-copy continuation errors under deletion.

D Additional Results for Experiment 3

D.1 Step segmentation

We split each human rationale into a sequence of discrete steps using a sentence/line-based splitter. We first normalize whitespace and split by line breaks, then further split long lines into sentences by punctuation. We drop empty fragments and keep the remaining fragments in order as the step sequence. All prefix truncations and window deletions operate on this step sequence. Let $L(e)$ denote the number of steps of instance e after segmentation.

D.2 Inference interfaces and prompts

We evaluate three inference interfaces (*policies*) on the GSM8K test set with human-written rationales:

No-CoT. The prompt contains the question only, and the model outputs `Answer: <integer>`.

EXTRACT. The prompt contains only the (possibly truncated) rationale steps and excludes the question. This interface tests whether the model can *read off* the final numeric answer from the visible rationale.

REPAIR. The prompt contains the question and the (possibly truncated) rationale steps, allowing the model to re-solve if needed.

All GSM8K runs use the same model as Exp. 1–2 with greedy decoding (τ=0).

D.3 Evaluation protocol and denominators

Support constraint for EXTRACT. To prevent EXTRACT from degenerating into re-solving, we impose a support constraint: a prediction is counted correct only if (i) the numeric answer matches gold, and (ii) the predicted number appears in the *visible*

Policy	Condition	MAIN	SOCRATIC
No-CoT	Question only	0.157	0.157
Extract	Full rationale (k=0)	0.988	0.992
Repair	Full rationale (k=0)	0.999	0.999

Table 4: Full-set (unconditioned) baseline accuracies on 1,000 GSM8K test instances per rationale style.

(kept) rationale steps. Moreover, if a deletion removes all occurrences of the gold answer from the kept steps, the extraction task becomes ill-posed; we exclude such instances from the evaluated set for that condition.

Used subsets (conditioning on full correctness). To isolate deletion effects rather than baseline failures, we report truncation and flip-rate results on policy-specific *used* subsets that condition on being correct under the full rationale (k=0). Formally, for rationale style $R \in \{\text{MAIN}, \text{SOCRATIC}\}$ and policy $P \in \{\text{EXTRACT}, \text{REPAIR}\}$, let

$$B_{R,P} = \{e : \hat{y}^{(\text{full})}_{R,P}(e) = y^*(e)\}.$$

For prefix deletion with $k > 0$, we further require sufficient remaining steps, and (for EXTRACT) well-posedness under the support constraint.

Window deletions and flip rates (ELD). For deleting a contiguous window starting at step index i with length w, we evaluate on

$$U_{R,P}(i, w) := \{e \in B_{R,P} : L(e) \geq i + w - 1\}.$$

For EXTRACT, we additionally restrict to instances where the gold answer $y^*(e)$ still appears in the kept steps (support constraint). We report the conditional flip rate

$$\text{flip}_{R,P}(i, w) := \Pr\big[\hat{y}^{(-i:i+w-1)}_{R,P} \neq y^*\big],$$

where the probability is taken over $e \in U_{R,P}(i, w)$, so denominators vary with (i, w).

D.4 Full-set baselines (unconditioned)

Table 4 reports accuracies on the full 1,000-instance GSM8K test set per rationale style *before* conditioning on full-rationale correctness. These numbers quantify the "coverage" of the used subsets.

D.5 Prefix truncation results on used subsets

This subsection reports the prefix-truncation accuracies used to summarize Exp. 3 under policy-specific used subsets. For each rationale style and policy, we condition on being correct under the full rationale ($k = 0$), and then evaluate how accuracy changes after deleting the first k steps. For EXTRACT, denominators may further shrink because we require the gold answer to remain supported by the visible rationale after truncation. These subset rules are the reason why the $k = 0$ rows are 1.000 by construction and why denominators vary across conditions.

Policy	Condition	MAIN	SOCRATIC
No-CoT	Question only	0.157	0.157
Extract	Full rationale (k=0)	1.000	1.000
Extract	Prefix-del (k=1)	0.997	0.998
Extract	Prefix-del (k=2)	0.993	0.989
Extract	Prefix-del (k=3)	0.976	0.991
Repair	Full rationale (k=0)	1.000	1.000
Repair	Prefix-del (k=1)	0.865	0.883
Repair	Prefix-del (k=2)	0.819	0.846
Repair	Prefix-del (k=3)	0.765	0.798

Table 5: Overall accuracy for Exp. 3 on *used* subsets. For each rationale style and policy, we condition on full-rationale correctness (k=0), hence the k=0 rows are 1.000 by construction. For $k > 0$, denominators shrink because we further require (i) at least k+1 steps remain after deletion, and (ii) for EXTRACT, the gold answer still appears in the kept steps (support constraint well-posedness).

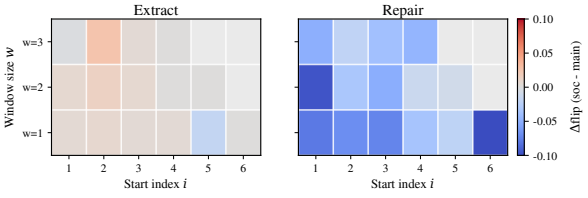

Figure 8: Window-deletion sensitivity heatmap: $\Delta\text{flip} = \text{flip}_{\text{SOCRATIC},P} - \text{flip}_{\text{MAIN},P}$. Negative values indicate SOCRATIC is more robust.

D.6 Window-deletion sensitivity (heatmap)

Figure 8 visualizes the SOCRATIC−MAIN difference $\Delta\text{flip} = \text{flip}_{\text{SOCRATIC},P} - \text{flip}_{\text{MAIN},P}$ for $w \in \{1, 2, 3\}$ and start indices i. We restrict the heatmap to early start indices (up to i=6) for readability, since denominators beyond that point become small and estimates are noisier (Table 6).

E Extended Camera-Ready Validations

E.1 Additional Models and Harder Benchmarks

To test whether the Extract–Repair separation generalizes beyond GSM8K and the two main mod-

i	$w = 1$ (delete step i)				$w = 2$ (start at i)				$w = 3$ (start at i)			
	Extract		Repair		Extract		Repair		Extract		Repair	
	M	S	M	S	M	S	M	S	M	S	M	S
1	0.000	0.005	0.224	0.146	0.001	0.008	0.331	0.235	0.004	0.004	0.350	0.300
2	0.262	0.269	0.212	0.144	0.395	0.405	0.256	0.223	0.485	0.504	0.307	0.287
3	0.378	0.380	0.193	0.120	0.478	0.480	0.265	0.215	0.562	0.557	0.289	0.251
4	0.456	0.456	0.213	0.178	0.558	0.549	0.247	0.234	0.600	0.596	0.300	0.255
5	0.549	0.523	0.188	0.167	0.571	0.569	0.245	0.236	0.738	0.682	0.289	0.267
6	0.581	0.578	0.300	0.200	0.714	0.659	0.156	0.133	0.818 †	0.692 †	0.214 †	0.214 †
7	0.738	0.682	0.156	0.133	0.909 †	0.846 †	0.143 †	0.214 †	0.500 †	0.667 †	0.000 †	0.000 †
8	0.909 †	0.846 †	0.143 †	0.286 †	0.500 †	0.667 †	0.000 †	0.000 †	0.000 †	0.000 †	0.000 †	0.000 †
9	0.500 †	0.333 †	0.000 †	0.000 †	0.000 †	0.000 †	0.000 †	0.000 †	1.000 †	1.000 †	0.000 †	0.000 †

Table 6: Window-deletion flip rates on $U_{R,P}(i, w)$ (varying denominators). M=MAIN, S=SOCRATIC. For $w = 1$, i denotes the deleted step index; for $w \in \{2, 3\}$, i denotes the start index. † indicates cells with denominator < 20 (very small evaluated sets), so these entries should be interpreted cautiously. We show $i \leq 9$ for readability; later indices have extremely small denominators.

els in the paper, we additionally evaluate stronger and newer model variants on harder mathematical benchmarks. The goal of these experiments is not to replace the main controlled analysis, but to test whether the same qualitative pattern persists under broader settings: robustness under EXTRACT, and substantially larger degradation under REPAIR when early rationale steps are removed.

Across these additional settings, the main qualitative pattern remains unchanged: deleting early steps has relatively limited effect under EXTRACT, where the model can often read off the answer from the remaining rationale, but causes much larger degradation under REPAIR, where the model must re-solve from incomplete intermediate context.

E.2 Budgeted-Continuation Control for the Prefix-Sum Plateau

A key concern is that the low-budget plateau in Scheme A might be an artifact of forbidding additional reasoning after truncation. To test this directly, we add a budgeted-continuation control: after exposing only the first few visible steps, we allow the model to continue generating reasoning up to the total token budget of a full trace. If the original plateau were mainly caused by the no-new-steps constraint, accuracy should recover substantially under this setting. Empirically, it does not, supporting the interpretation that the plateau reflects missing computational state rather than a purely procedural stopping artifact.

E.3 Format-Preserving and Random-Drop Perturbation Controls

Because step deletion may introduce out-of-distribution formatting artifacts, we compare standard prefix deletion against format-preserving masking and random dropping controls. Prefix masking preserves sequence length and coarse positional structure while removing early information, helping disentangle missing computation from formatting disruption. The results show that formatting does matter, but cannot fully explain the degradation: even under format-preserving masking, REPAIR degrades substantially, whereas EXTRACT remains much more robust.

Thus, the degradation observed under REPAIR cannot be reduced to formatting shift alone; missing intermediate computational state remains the dominant factor.

This result suggests that the required sequential depth of the task does not disappear, but may be realized differently across model variants. In particular, the think variant appears able to internalize more of the required sequential computation into latent states, reducing its dependence on long visible CoT traces.

E.4 Unconditioned Evaluation and Statistical Significance

Several analyses in the main text condition on full-rationale correctness in order to isolate deletion effects. To quantify the resulting bias, we additionally report unconditioned evaluations on the full dataset and compare the resulting degradation trends against the used-subset analyses. We further report paired significance tests for the early-step degradation under REPAIR. Although conditioning raises absolute accuracies, the relative Extract–Repair contrast and the early-deletion degradation pattern remain unchanged.

Conditioning on full-rationale correctness raises

Model	Data	Full Ext.	Trunc-$k=3$ Ext.	Full Rep.	Trunc-$k=3$ Rep.	ΔRep.
Llama-3.1-8B-Instruct	MATH	0.944	0.917	0.940	0.684	-0.256
Qwen2.5-7B-Instruct	MATH	0.932	0.947	0.952	0.894	-0.058
Gemma-2-9B-IT	MathQA	0.880	0.825	0.912	0.765	-0.147
Llama-3.1-8B-Instruct	MathQA	0.884	0.805	0.918	0.624	-0.294

Table 7: Additional results on harder benchmarks. Ext.=EXTRACT, Rep.=REPAIR, and ΔRep. denotes Trunc-$k=3$ minus Full under REPAIR.

Protocol	Condition	Accuracy
Truncate-Stop	No new tokens	0.0175
Budgeted-Cont.	Generates freely	0.0152
No-CoT	Zero-shot	0.0080

Table 8: Budgeted-continuation control on the depth-32 prefix-sum task in the low-visible-budget regime. Allowing the model to continue generating reasoning does not substantially recover accuracy relative to the original Truncate-Stop setting: Budgeted-Cont. remains close to the Truncate-Stop baseline and only slightly above the No-CoT floor.

If.	Intv.	Fmt.	Acc.	Δ
Ext.	Full	Y	0.968	—
Ext.	mask	Y	0.969	+0.001
Ext.	rand-drop	N	0.504	-0.464
Rep.	Full	Y	0.883	—
Rep.	mask	Y	0.830	-0.053
Rep.	delete	N	0.702	-0.181

Table 9: OOD-control ablations on GSM8K-Main at $k=2$. If.=interface, Intv.=intervention, Fmt.=format preserved, and Δ is accuracy change relative to the full baseline.

absolute accuracies, but does not alter the qualitative Extract–Repair contrast or the existence of a statistically significant early-step degradation effect.

E.5 Sampling vs. Greedy Decoding

We additionally tested non-zero-temperature sampling on GSM8K to assess whether the observed Time-for-Accuracy and deletion-sensitivity trends depend on greedy decoding. While sampling changes the absolute accuracy of individual trajectories, the qualitative pattern remains the same: the Extract–Repair separation persists, and early-step deletion remains substantially more damaging under REPAIR. We therefore use greedy decoding in the main paper to isolate the depth limit of a single trajectory without introducing additional search breadth.

Variant	Visible steps	Acc.	Finding
Qwen3-8B (no-think)	$T=2$	0.000	Collapse
Qwen3-8B (think)	$T=2$	0.687	Latent reason

Table 10: Qwen3-8B with and without latent thinking under extreme visible-trace truncation on the depth-32 prefix-sum task.

Data	Subset	ΔAcc	p
GSM8K-Main	All	-0.053	< 0.01
GSM8K-Main	Used	-0.109	< 0.01

Table 11: Significance test for early-step truncation under REPAIR. ΔAcc = Trunc − Full.

F Extended Related Work

CoT, sampling, and inference-time scaling. CoT prompting was introduced as providing step-by-step exemplars and later shown to work even with simple zero-shot triggers (Wei et al., 2022; Kojima et al., 2022). Self-consistency improves CoT by sampling diverse reasoning paths and marginalizing over final answers (Wang et al., 2022). More broadly, test-time compute scaling studies how performance depends on inference compute (samples, steps, search) under fixed budgets (Wu et al., 2024; Snell et al., 2024; Zhang et al., 2025a). In domains with strong verification signals, ranking and verification of trajectories can yield substantial gains (Cobbe et al., 2021; Chen et al., 2024; Lu et al., 2024; Wang et al., 2024b); without verifiers, selection and aggregation may plateau despite additional compute (Brown et al., 2024). Our work studies a complementary axis: instead of distributing compute across many candidate trajectories, we characterize how a single chain's budget T interacts with the required *sequential depth* of the task, predicting plateau-to-transition curves.

Step-level evaluation, PRMs, and how ELD differs. Step-level benchmarks aim to localize errors and assess intermediate reasoning quality, e.g.,

by identifying earliest incorrect steps (Zheng et al., 2025). PRMs and related process-supervised approaches provide learned signals that score steps or trajectories and can support guided selection/search (Khalifa et al., 2025; Wang et al., 2025). In retrieval/agentic settings, process-level rewards have also been used to train multi-step policies beyond outcome-only signals (Zhang et al., 2025b). These approaches typically rely on explicit supervision or a separate learned evaluator. By contrast, our Effective Logical Depth (ELD) is an *intervention-based* proxy: we delete parts of a trace and quantify how often correctness breaks under a specified interface. This makes ELD protocol-dependent (e.g., Extract vs. Repair), but also allows it to directly estimate *causal necessity* of steps without requiring step labels or a verifier model.

Faithfulness, robustness, and controlled analyses of CoT. As CoT becomes ubiquitous, work on long-CoT and reasoning trace evaluation highlights phenomena such as overthinking, brittleness, and the difficulty of interpreting traces as faithful computation (Chen et al., 2025). Controlled studies investigate why CoT helps and when it fails, attributing gains to factors such as probability effects, memorization, or noisy reasoning in specific environments (Prabhakar et al., 2024). In planning, CoT improvements can be prompt-specific and may degrade with increasing complexity (Stechly et al., 2024). Related robustness concerns appear in broader generation settings such as hallucination detection (Benkirane et al., 2024). Our deletion-based experiments complement these lines by explicitly separating *reading off* from *recomputing*: under Extract the model may rely on suffix cues, while under Repair it is forced toward sequential recomputation, making early deletions more damaging.

Algorithmic-information view: description length vs. sequential effort. Kolmogorov complexity formalizes information content as shortest description length, while Bennett's logical depth captures the runtime of near-shortest programs, measuring sequential effort required to generate an object (Li and Vitányi, 2008; Bennett, 1988). Recent work connects such complexity notions to instance difficulty and compute allocation, e.g., via compression-style proxies that approximate complexity and guide compute (Yoran et al., 2025). We instead use *relative logical depth* to motivate a qualitative Time-for-Accuracy prediction: when chain budget is below the required depth, a depth-induced error plateau persists; once budget matches required depth, the computational floor disappears. Deletion-based ELD then provides an operational bridge between this depth requirement and observable trace behavior, distinguishing raw length from realized sequential computation.