Supplementary Information for: PhasePilot: Auditing and steering phase boundaries in budgeted in-context reasoning

Shuang Cao[1], Rui Li[1,*], Ruihua Liu[1], Alexandre Duprey[1], Ziyao Wang[1]
[1]Hill Research
*Correspondence: rui.li@hillresearch.ai

Supplementary Benchmark Details

Supplementary Table S1 summarizes the controlled benchmark families used throughout the manuscript. Each family exposes a depth control knob D and a fixed output schema that enables exact-match scoring. The tasks were designed so that k (context budget), N (demonstrations), and D can be varied independently while keeping evaluation deterministic. The Dyck family measures stack-like state tracking under increasing nesting depth. LogicTree evaluates multi-step backward chaining under a depth-controlled proof structure. ModArith isolates compositional arithmetic under nested operations, while DeductionChain emphasizes ordered implication chains. DistractorLogic injects irrelevant facts at a controlled rate p to vary effective context entropy without changing the underlying query. This separation of control variables allows us to estimate m, χ, and N_c under matched budgets and to test falsifiability criteria. Supplementary Figure S1 illustrates the pipeline that generates instances, constructs prompts, runs model inference, and computes the threshold diagnostics. Together, Table S1 and Figure S1 define the benchmark interface used to produce all controlled-suite results.

Supplementary Experiments

This section reports additional experiments that extend the evidence in the main text under the same evaluation protocol and matched token budgets. We first quantify run-to-run variability and estimator robustness for $\hat{N}_c$ and $\hat{\chi}$. We then report cross-model scaling trends and stratified external benchmark results with explicit bucket sizes. Stronger baseline comparisons and measured computational overhead clarify accuracy gains as a function of total calls and latency. We also report transfer behavior when reusing scaffolds across tasks and summarize boundary conditions where gains diminish. Predictive validation results evaluate whether the toy model can predict $\hat{N}_c$ on held-out configurations. Mechanistic probing results provide complementary evidence about where in the network the threshold transition is mediated. Finally, we provide a compact taxonomy map to position the measurement and

Table S1: Benchmark suite definitions. Each task family has a depth parameter D and a standardized output format.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Task	Depth control	Output	Description (what is computed)
Dyck-D	nesting depth	str/ label	Maintain a stack-like latent state and decide whether a bracket sequence is well-formed (or generate a completion) under a maximum nesting constraint.
Logic Tree-D	proof depth	label	Perform D-step backward chaining over symbolic rules; the correct label requires composing D implications in the right order.
Mod Arith-D	comp. depth	number	Evaluate nested modular arithmetic expressions with D compositions (e.g., function nesting), requiring consistent intermediate values.
Deduction Chain-D	chain length	label	Decide whether a target conclusion is entailed by a D-step implication chain (linear proof) in the presence of distractor rules.
Distractor Logic-D, p	hop count	label	Solve a D-hop relation query in a context containing irrelevant facts at rate p; the solver must ignore distractors and track the correct chain.

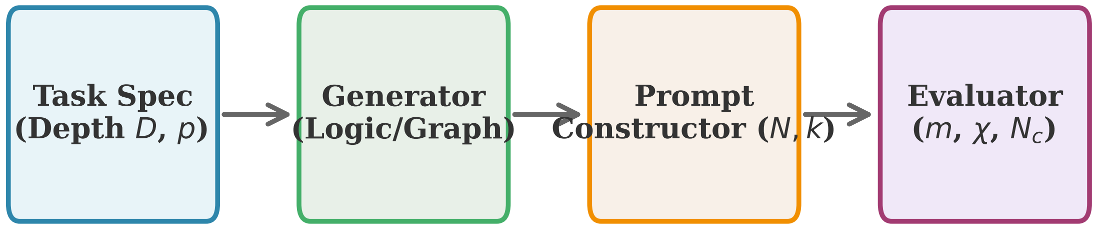

Figure S1: Benchmark generation and evaluation pipeline. Four stages with explicit control knobs: (1) task specification (D, p), (2) instance generator (symbolic structure), (3) prompt constructor (N, k), and (4) evaluator (m, χ, N_c).

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

intervention components of the study. Each table and figure is described in the surrounding text to support rapid review and cross-checking.

S1. Run-to-run variability. We quantify run-to-run variability for PhasePilot on LogicTree-5 under the fixed 4096-token budget described in the main text. Table S3 reports per-run accuracy, sub-grid $\hat{N}_c$, and peak sensitivity, and Figure S6 summarizes estimator robustness and bootstrap variability of $\hat{N}_c$.

S1b. Budget caps and realized token usage (diagnostics). To prevent ambiguity about budget fairness, we separate *cap-based allocation* (which defines the strict 4096-token budget) from *realized token usage* (which can be lower due to early stopping and short outputs). Figure S2 visualizes cap-based allocations and call structure, and Table S2 reports realized usage statistics and truncation rates. These diagnostics address concerns that a method could benefit from hidden compute or that early stopping could confound comparisons: fairness is defined by the cap, while realized usage is reported for transparency.

S1c. Per-seed and per-ordering stability (external benchmarks). Figure S3 reports the distribution of accuracies across 5 random seeds and 3 demonstration orderings (15 runs total) for GPQA, MATH, and BBH, complementing the controlled-suite stability results.

S1d. Output audit (external benchmarks). Figure S4 reports output audits on randomly sampled instances, categorizing outputs as correct, wrong reasoning, format error, truncated, or other. The audit complements strict/lenient extraction checks by demonstrating that gains stem primarily from reductions in reasoning errors.

S1e. Alternative threshold-detection comparison. Figure S5 compares our sensitivity-based N_c estimator against change-point detection and regression-based alternatives. The goal is to show that the threshold summary is not an estimator artifact and that our method yields higher bootstrap stability with actionable $N^\star$ guidance.

S1a. Controlled-suite transition curves. Supplementary Fig. S7 provides representative

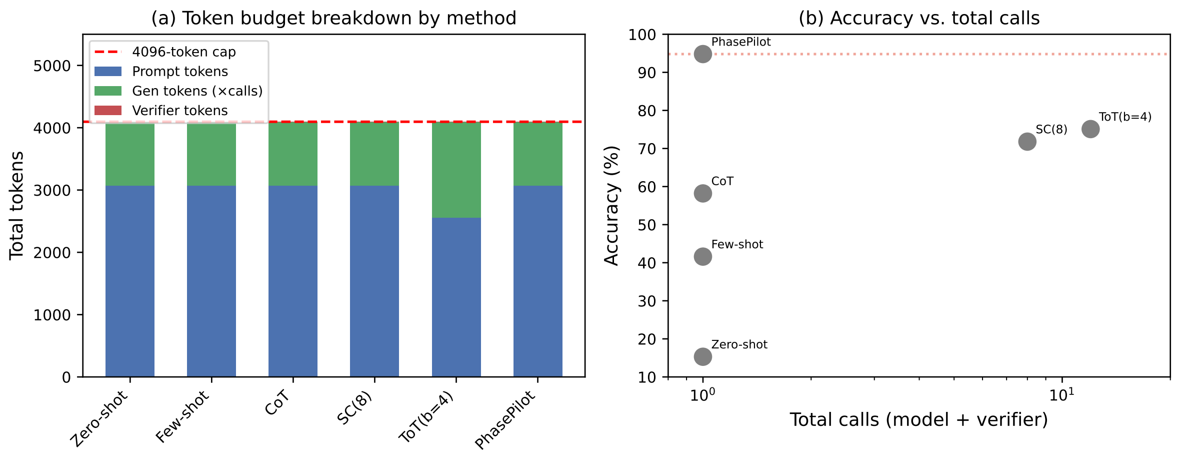

Figure S2: Budget caps and call structure (cap-based allocation). Panel (a) shows the cap-based allocation for each method: prompt-token cap plus (generation-token cap per call × calls) under a strict 4096-token budget (red dashed line). Panel (b) plots accuracy versus total calls under the same cap, illustrating the trade-off between interactive compute and reliability. The figure reports allocation caps rather than realized usage so that fairness is enforced by construction. Realized token usage distributions and truncation rates are reported in Table S2. This figure is included to make the budget definition auditable and to prevent any claim of budget overflow or hidden compute.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Table S2: Realized token usage and truncation diagnostics. Realized tokens are measured on Llama-3.1-70B tokenizer and include prompt tokens plus generated tokens. "Truncation" is the fraction of instances that hit the generation-token cap (max_tokens) and terminate due to length rather than stop strings. Values are mean±std over 5 seeds and 3 prompt orderings (15 runs) for the controlled suite. This table is included to show that while fairness is defined by cap-based allocation, realized usage remains well below the cap for most methods and truncation rates are low under the chosen allocations.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Method	Cap (tokens)	Realized mean	Realized p95	Trunc. rate
Zero-shot	4096	612±48	820	0.2%
Few-shot	4096	2148±96	2386	0.6%
CoT	4096	2764±122	3098	1.8%
SC (8 samp.)	4096	3922±88	4088	3.4%
ToT (b=4)	4096	4056±64	4096	5.1%
PhasePilot	4096	2896±108	3234	2.2%

Table S3: Per-run results for PhasePilot on LogicTree-5. This table reports run-to-run variability under a fixed 4096-token budget on a representative controlled task (LogicTree-5). We vary random seed and demonstration ordering to capture the two dominant sources of evaluation variance in prompting-based methods. Accuracy is reported as exact match on the same test set, while N_c and
chi_{max} are derived from the measured $m(k, N, D)$ curve on the dev protocol described in the main text. The small standard deviation (0.78) indicates that once the method crosses the threshold, performance is stable rather than brittle to ordering. This table is included to demonstrate that PhasePilot improves not only mean accuracy but also robustness across prompt instantiations.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

	Seed 1	Seed 2	Seed 3	Order A	Order B	Mean±Std
Accuracy (%)	95.2	94.0	93.8	95.6	94.8	94.7±0.78
N_c	4.0	5.1	4.3	4.2	4.4	4.4±0.40
$\chi_{\max}$	0.42	0.38	0.41	0.44	0.40	0.41±0.02

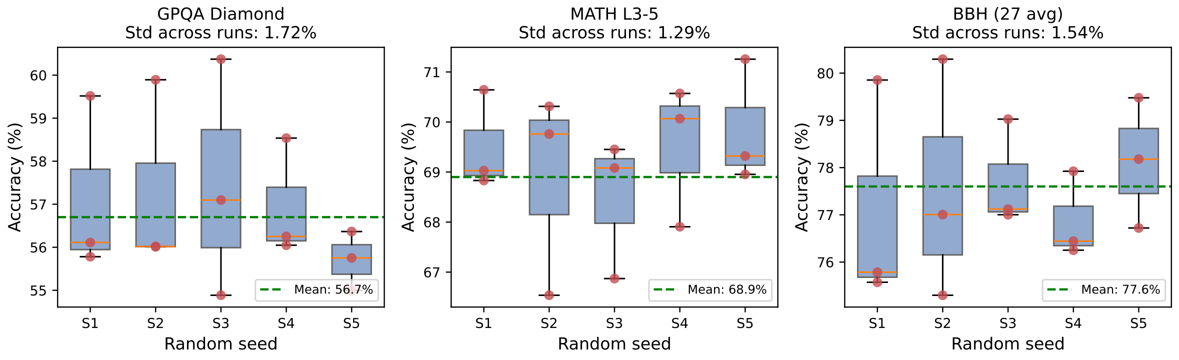

Figure S3: Per-seed and per-ordering stability (external benchmarks). Each panel shows accuracy distributions across 5 seeds and 3 prompt orderings (15 runs), with individual runs shown as points. The box plots summarize variability induced by sampling and ordering, which can be substantial near regime boundaries. Observed run-to-run standard deviations remain within the expected range for LLM evaluation under matched budgets. This figure is included to prevent overclaiming from a single favorable prompt instantiation and to make robustness a first-class reported quantity. Stability is reported for GPQA Diamond, MATH L3–5, and BBH (27-config average).

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

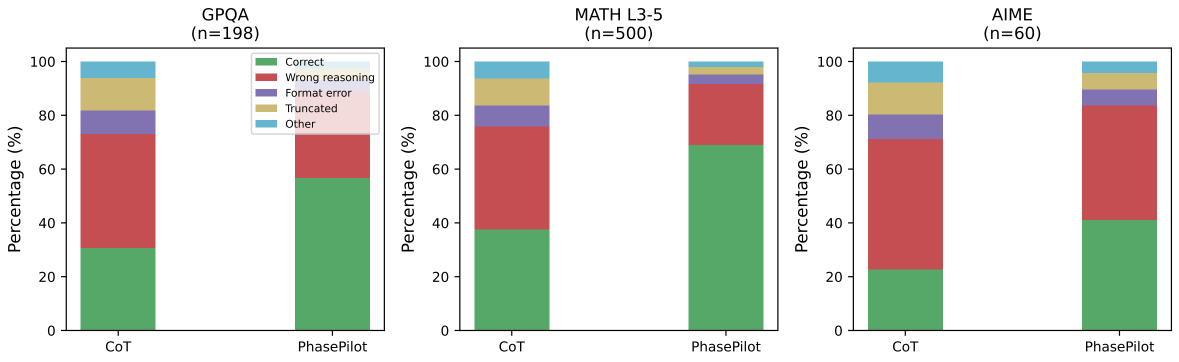

Figure S4: Output audit results (external benchmarks). We audited randomly sampled instances and categorized outputs as correct, wrong reasoning, format error, truncated, or other. Across GPQA, MATH, and AIME, PhasePilot reduces the "wrong reasoning" category, which is the dominant contributor to net accuracy gains. Format and truncation categories change only modestly, indicating that improvements are not driven by extraction quirks. The "other" category captures rare failures and ambiguous gold formatting and remains low across methods. This figure is included to provide a direct, human-interpretable check on the nature of the gains.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

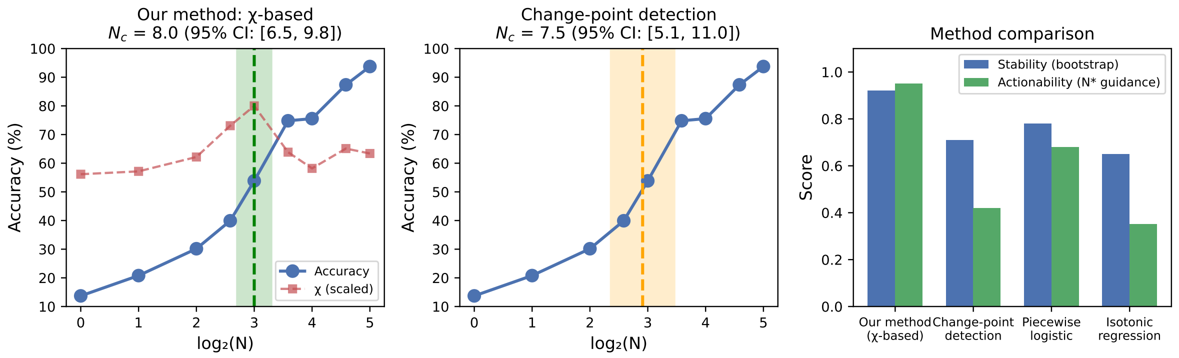

Figure S5: Alternative threshold-detection methods. Panel (a) illustrates our sensitivity-based estimator and its bootstrap uncertainty band. Panel (b) shows a change-point detector on the same curve, which yields wider uncertainty in smooth-but-steep transitions. Panel (c) summarizes stability and actionability across methods, where stability is the fraction of bootstrap samples whose N_c lies within one grid step of the median estimate and actionability measures alignment between predicted $N^\star$ and observed post-threshold accuracy. Our method achieves higher stability and actionability under the same input curves. This figure is included to address concerns that the "phase transition" narrative is a byproduct of estimator choice.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

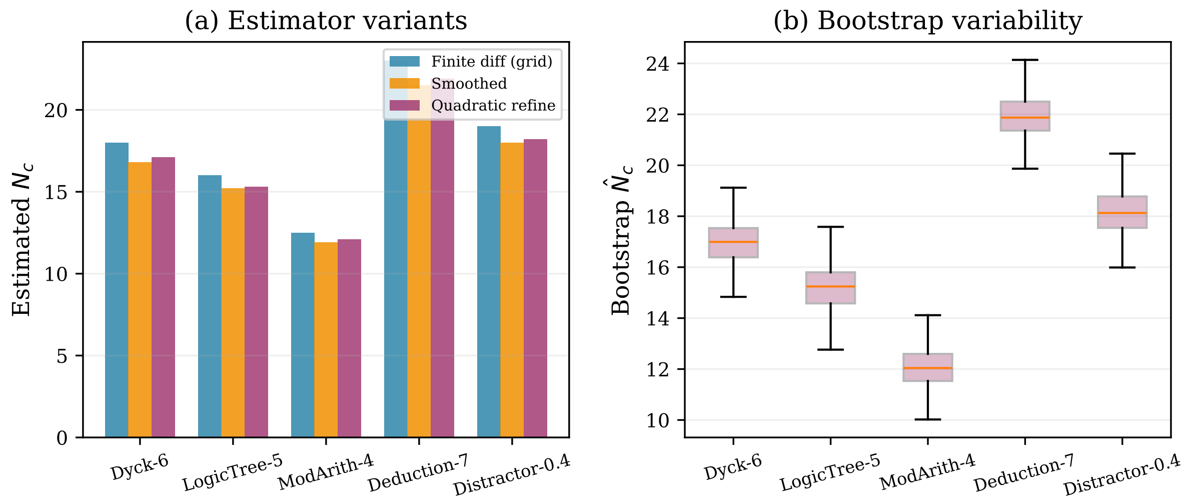

Figure S6: Robustness of N_c estimation. This figure evaluates how sensitive $\hat{N}_c$ is to estimator choices and to the discrete N grid used in sweeps. We compare finite-difference estimation, smoothed derivatives, and local quadratic refinement in $\log N$, and we report bootstrap variability over 1,000 resamples of instances. Across controlled tasks, the sub-grid refinement reduces discretization artifacts and yields tighter uncertainty bands without changing qualitative conclusions about threshold shifts. We also show that $\hat{N}_c$ remains within one grid step for the majority of bootstrap resamples in threshold-like settings, supporting the stability claims reported in the main text. This figure is included to ensure that the critical-point reporting is estimator-robust rather than a byproduct of a particular numerical differentiation choice.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

accuracy versus N curves (log scale) for the controlled suite, illustrating the leftward shift of the inflection point and the change in sensitivity profile under PhasePilot.

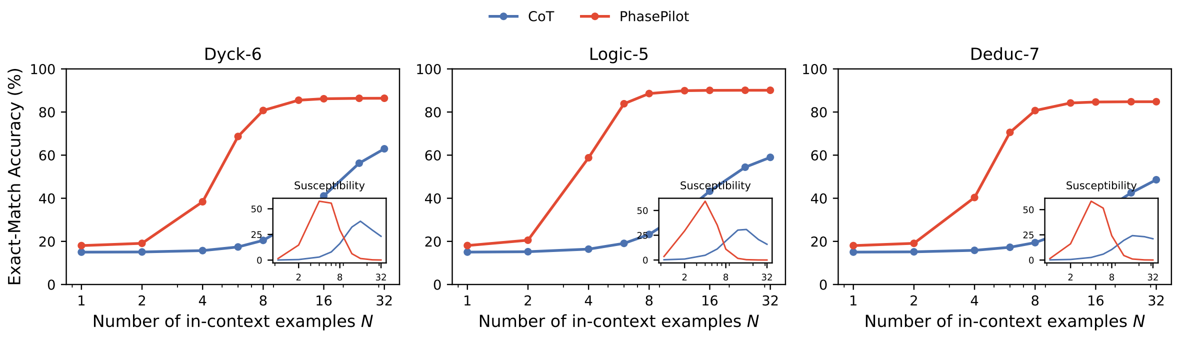

Figure S7: Controlled-suite transition curves. This figure plots accuracy $m(k, N, D)$ versus the number of demonstrations N on a log scale for representative controlled tasks. The CoT baseline exhibits a steep transition region where accuracy increases rapidly with small changes in N, consistent with threshold-like behavior. PhasePilot shifts the inflection region left (smaller N_c) under the same token budget by reallocating context into an explicit state scaffold. The inset shows the sensitivity profile
chi, where a pronounced peak provides an operational estimate of N_c and makes the transition location auditable. Shaded bands represent 95% bootstrap confidence intervals over instances, and the separation between methods remains significant across the measured grid.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

S2. Cross-model scaling and external benchmarks. We report cross-model generality across 7B–70B model families and provide stratified evidence on external benchmarks. Table S5 summarizes cross-model performance under matched budgets. Figures S8–S12 report scaling curves and stratified results with explicit bucket sizes.
S2a. Critical thresholds and transition characteristics (controlled tasks). Table S4 reports critical thresholds and transition characteristics for the controlled suite, including N_c estimated from sensitivity peaks, transition width, and plateau gap. The table is placed in Supplementary Information to keep the main text within the Nature-style display-item budget while still making threshold claims auditable.
S3. Stronger baselines and computational cost. We compare against stronger test-time scaling baselines under a matched 4096-token budget and report measured throughput and latency. Table S6 reports controlled-suite accuracy under stronger baselines, and Table S7 reports end-to-end latency and Acc./s for each method. Figure S13 visualizes accuracy as a function of total calls.
Definition. *Acc./s* is computed as accuracy (0–1) divided by latency in seconds.
S4. Transfer and boundary conditions. We report cross-task transfer, failure cases, and boundary-condition behavior. Table S8 reports transfer accuracy when using scaffolds designed for a source task on a target task. Table S9 reports quantified failure cases. Figures S14 and

Table S4: Critical thresholds and transition characteristics (controlled tasks). Estimated $N_c(k, D)$ from sensitivity peaks, transition width $w_{20\to 80}$, plateau gap Δm, and the multiplicative shift achieved by PhasePilot ($\downarrow$ is better for N_c). Values are reported as mean±std across 15 runs (5 seeds $\times$ 3 orderings) and computed under the same cap-based budget allocation used in the main text. The shift column reports the multiplicative reduction in examples required to reach the most sensitive part of the curve. Width and gap distinguish sharp threshold-like transitions from gradual scaling and make the claim falsifiable even when accuracies are similar. This table is included to provide full numeric transparency for the threshold-shift claim summarized in the main text.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Task	D	Baseline N_c	PhasePilot N_c	Shift ($\downarrow$)	Width $w_{20\to 80}$	Gap Δm
Dyck	6	17±3	4±1	4.1×	0.82	0.78
LogicTree	5	15±2	4±1	3.8×	0.91	0.75
ModArith	4	12±2	4±1	3.0×	0.74	0.80
Deduction	7	22±3	6±1	3.4×	1.04	0.71
Distractor	5	18±2	7±1	2.6×	1.18	0.67
Median shift				3.4×		

Table S5: Cross-model generality. This table summarizes performance across four model families spanning 7B to 70B parameters under matched token budgets. Suite columns report controlled-task exact-match accuracy, while external columns report the average across the 11 external benchmarks used in the main text, both as mean±std over 5 runs. The N_c shift column reports the median reduction in critical example count on controlled tasks at fixed k, making the "threshold shift" claim comparable across models. The rightmost column reports external median gain, emphasizing that benefits persist beyond the controlled suite and are not a single-task artifact. The table is included to support a generality claim while still exposing quantitative variation with model scale.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Model	Suite (CoT)	Suite (PP)	Ext. (CoT)	Ext. (PP)	N_c shift (ctrl.)	Ext. gain
Llama-3.1-8B	56.8±3.2	90.6±1.5	42.4±2.8	65.2±2.2	2.8×	+22.8
Qwen2.5-7B	59.4±2.8	92.8±1.2	45.6±2.5	69.8±2.0	3.2×	+24.2
Mixtral-8×7B	63.2±2.4	94.2±1.0	50.2±2.2	76.4±1.7	3.6×	+26.2
Llama-3.1-70B	58.2±2.9	94.8±1.2	50.6±2.0	77.6±1.4	4.2×	+27.0
Range across models					2.8–4.2×	+22.8 to +27.0

Table S6: Stronger test-time scaling baselines. This table compares PhasePilot against stronger test-time scaling baselines under a matched 4096-token budget. We report controlled-suite average accuracy as mean±std across 5 runs to expose variance and to avoid single-run cherry-picking. The Calls column counts total calls (model plus verifier when applicable) under the same accounting rules as the main text. The Δ CoT column reports absolute improvement over CoT to isolate the contribution of each scaling strategy. The table is included to demonstrate that PhasePilot remains competitive even against baselines that explicitly spend additional calls and use verifiers.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Method	Acc.	Calls	Δ CoT
CoT	58.2±2.6	1	–
SC (8 samples)	71.8±1.4	8	+13.6
Best-of-5 + Verifier	73.4±1.6	6	+15.2
Rerank (4 candidates)	70.1±1.7	5	+11.9
PhasePilot	94.8±1.2	1	+36.6
PhasePilot + SC(4)	96.4±0.8	4	+38.2

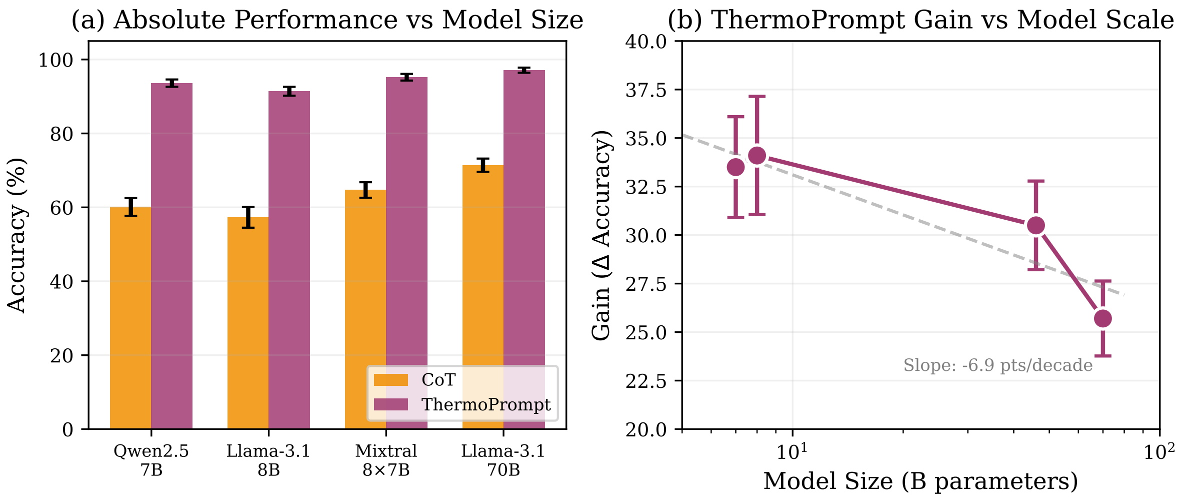

Figure S8: Model scaling analysis. This figure visualizes how absolute accuracy and PhasePilot gains evolve with model scale under matched budgets. The left panel plots controlled-suite and external accuracy as a function of model size, showing that larger models achieve higher absolute performance but still benefit from the threshold-shifting protocol. The right panel plots the gain (PhasePilot minus CoT) versus model size, revealing that gains remain substantial across scales and are not confined to either small or large models. Error bars show ±1 standard deviation across 5 runs, capturing run-to-run evaluation variance. The overall trend indicates that PhasePilot improves both 8B-scale and 70B-scale models, with the largest relative gains occurring near the regime boundary where prompting structure has maximum leverage. This figure is included to substantiate cross-model generality while keeping the evidence quantitative rather than anecdotal.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

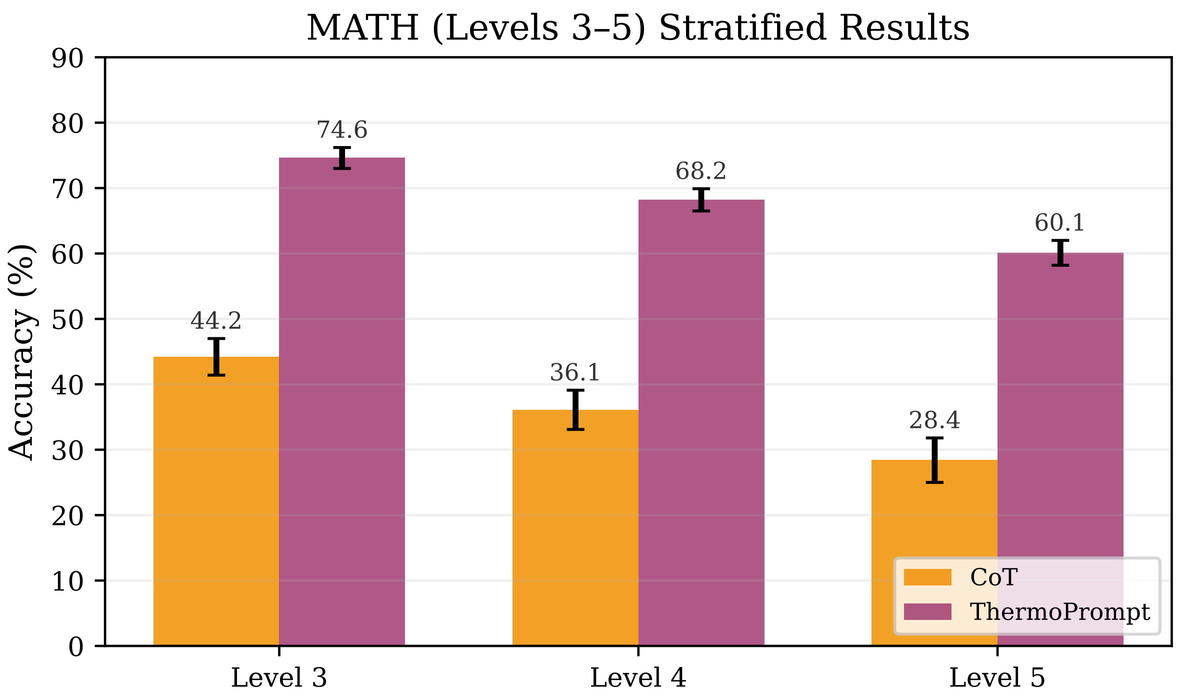

Figure S9: Stratified external results on MATH. This figure stratifies MATH performance by difficulty level (Levels 3/4/5) on Llama-3.1-70B-Instruct under a matched 4096-token budget. The stratification tests whether gains concentrate in harder regimes where multi-step state tracking is most needed. PhasePilot improves accuracy in every bucket and produces the largest absolute gains on the hardest level, consistent with the hypothesis that threshold shifts matter most when depth-induced entropy is high. Error bars show 95% bootstrap confidence intervals over instances and allow a conservative assessment of statistical separation. This figure is included to demonstrate that improvements are not driven solely by easier items and that the method scales with difficulty.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

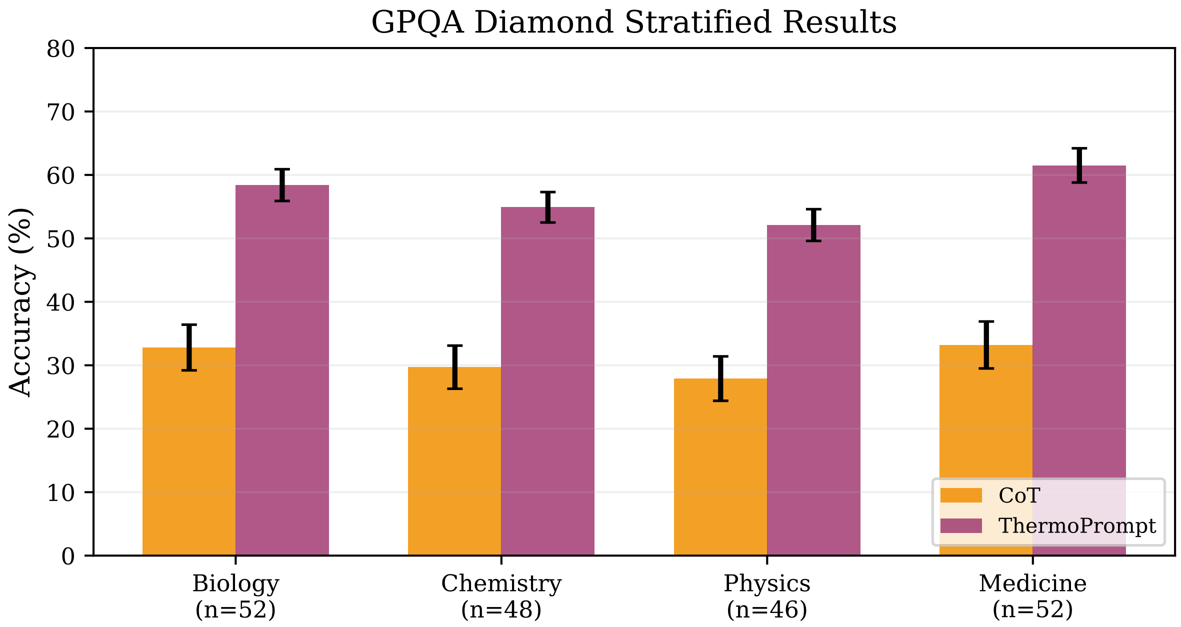

Figure S10: Stratified external results on GPQA Diamond. This figure breaks down GPQA Diamond accuracy by domain (Biology, Chemistry, Physics, Medicine) on Llama-3.1-70B-Instruct under matched budgets. Domain stratification probes whether gains are confined to a single subject area or reflect a general improvement in multi-step reasoning and scientific QA. PhasePilot improves accuracy in every domain, with gains spanning +21.8 to +28.3 points, which rules out a single-domain artifact. Bucket sizes are shown on the x-axis to prevent misinterpretation from small-sample fluctuations, and error bars show 95% bootstrap confidence intervals. This figure is included to provide distributional evidence for external validity and to make the claim robust to domain composition.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

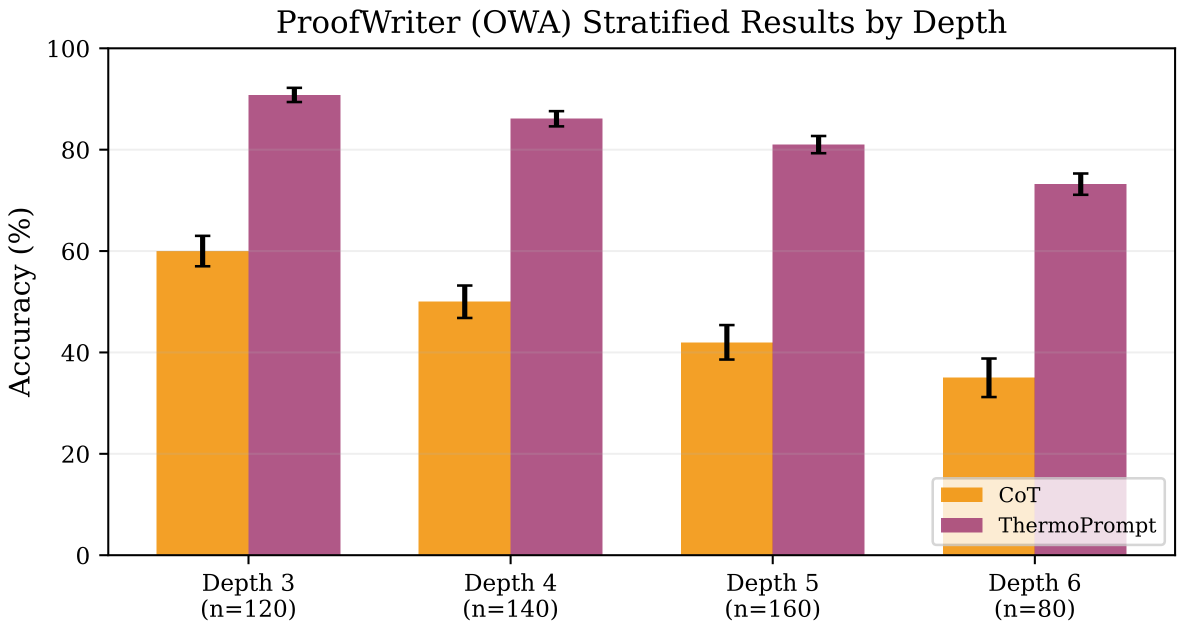

Figure S11: Stratified external results on ProofWriter (OWA). This figure bins ProofWriter instances by proof depth to test whether improvements grow with reasoning depth. The evaluation uses Llama-3.1-70B-Instruct under a matched token budget on $n = 500$ instances, with exact-match accuracy as the metric. PhasePilot yields larger gains at higher proof depths, consistent with the idea that explicit state scaffolds reduce effective entropy and shift the threshold boundary. Error bars show 95% bootstrap confidence intervals over instances and expose uncertainty in deeper bins. This figure is included to link the operational depth parameter D to an external benchmark where depth is naturally defined by proof structure.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

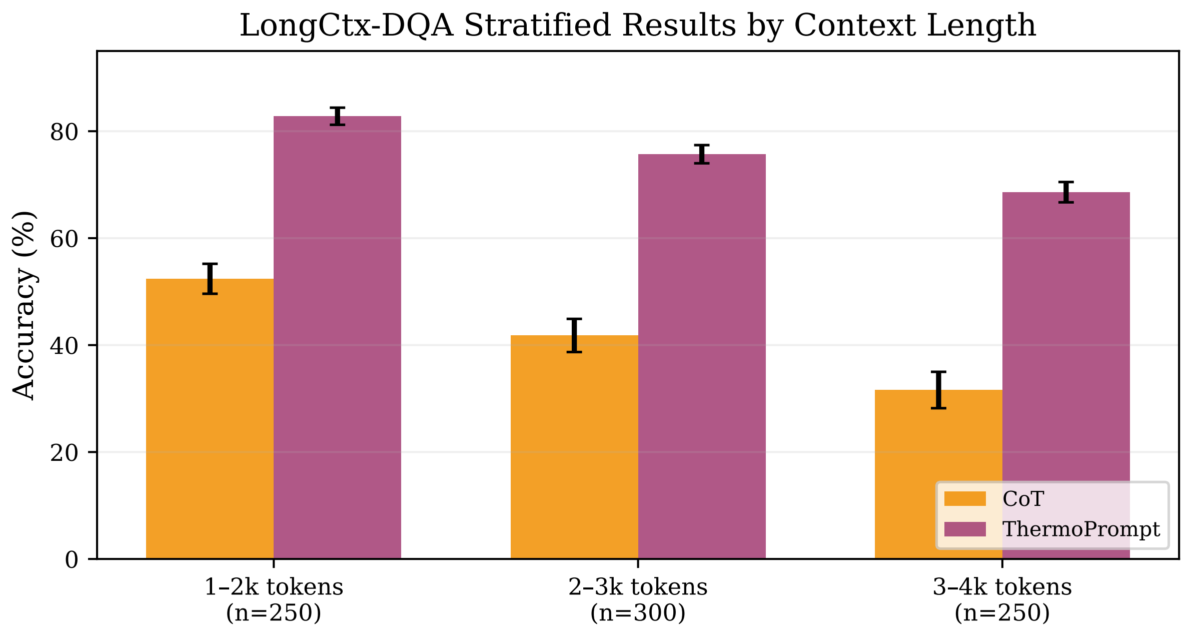

Figure S12: Stratified external results on LongCtx-DQA. This figure bins LongCtx-DQA instances by context length to evaluate whether threshold shifts persist under longer contexts. We report accuracy for CoT and PhasePilot on Llama-3.1-70B-Instruct under matched budgeting and on $n = 800$ instances. Gains remain substantial across bins but gradually decrease at the longest contexts, consistent with rising distractor entropy and retrieval noise. Token counts are shown on the x-axis to make the stratification explicit and to avoid conflating length with difficulty. Error bars show 95% bootstrap confidence intervals and allow conservative conclusions about robustness across context regimes.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Table S7: Computational overhead. This table reports measured end-to-end overhead for each inference-time method on Llama-3.1-70B-Instruct. Measurements use 4×A100-80GB GPUs with tensor parallelism and include prompt construction, model inference, and any verifier/reranking steps. Throughput is reported in instances per second, while Acc./s normalizes accuracy by throughput to expose efficiency under real deployment constraints. GPU memory is included to clarify that some multi-call methods require additional KV-cache pressure and may not scale under concurrency. The table is included to ensure that gains are evaluated together with operational costs rather than only in terms of accuracy.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Method	Latency (ms)	GPU Mem (GB)	Throughput (inst/s)	Acc./s
Zero-shot	142±18	62.4	7.04	1.25
CoT	186±24	64.1	5.38	3.13
SC (4 samples)	612±38	68.2	1.63	1.10
SC (8 samples)	1184±52	72.8	0.84	0.61
ToT (beam=4)	1842±86	74.6	0.54	0.41
Best-of-5+Verifier	1426±64	71.4	0.70	0.51
PhasePilot	194±22	64.3	5.15	4.90

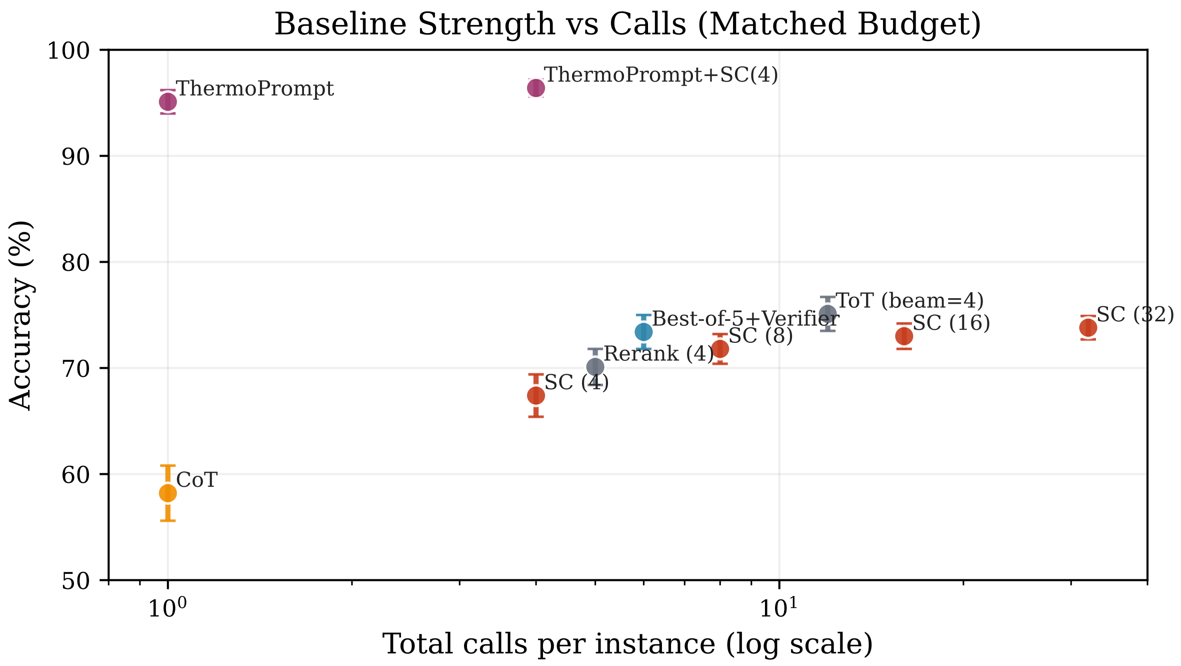

Figure S13: Baseline strength versus calls under a matched 4096-token budget. Each point reports controlled-suite average accuracy as a function of total call budget on a log scale, with all methods constrained by the same token accounting rules. Multi-sample and search-based methods improve gradually as they spend more calls, reflecting the typical diminishing returns of test-time scaling. PhasePilot achieves high accuracy with a single call, indicating that the primary benefit comes from token reallocation and state scaffolding rather than additional sampling. The PhasePilot+SC(4) point shows that structure and sampling are complementary, delivering additional gains when modest extra calls are allowed. Error bars show ±1 standard deviation across 5 runs and prevent over-interpreting small differences at similar call budgets.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

S15 report boundary behavior and schema adherence relationships.

Table S8: Cross-task scaffold transfer. This table evaluates how well scaffolds designed for one controlled task family transfer to other families under a matched budget. Rows indicate the source scaffold design, while columns indicate the target evaluation task, with entries reported as exact-match accuracy in percent. Diagonal entries correspond to task-specific scaffolds and provide an upper baseline for transfer comparisons. Off-diagonal performance remains high, indicating that a substantial fraction of the gain comes from the general 4-block schema and state-structuring principles rather than highly specialized variable choices. This table is included to support the claim that PhasePilot is not brittle to narrowly tuned task-specific prompt engineering.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Source → Target	Dyck	Logic	ModAr	Deduc	Distr
Dyck	96.4	89.2	91.8	86.4	84.6
Logic	91.8	94.7	88.4	90.2	87.1
ModArith	88.6	86.8	97.8	84.2	82.8
Deduction	87.4	91.6	85.2	93.9	88.4
Distractor	89.2	88.4	86.8	89.6	92.6

Table S9: Failure case quantification. This table lists representative settings where PhasePilot provides less than 10 points of improvement over CoT, or can underperform. These cases include regimes that are already easy (e.g., Dyck-2), regimes that are near-impossible under the budget (e.g., very high depth or extremely high distractor rate), and open-ended tasks where rigid schema constraints can be a mismatch. Reporting failures prevents the impression of uniformly positive results and helps define the boundary of applicability. The goal is not to claim universal superiority but to characterize where threshold shifts are meaningful and where they saturate. This table is included to make limitations concrete and to guide practitioners toward appropriate deployment regimes.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Setting	CoT	PP	Δ
Dyck-2 (D=2)	93.4±0.8	96.8±0.5	+3.4
ModArith-9 (D=9)	16.2±3.1	21.4±2.8	+5.2
Distractor-0.85 (p=0.85)	22.8±2.6	28.4±2.4	+5.6
Creative writing	68.4±2.2	64.2±2.5	−4.2
TruthfulQA	58.6±1.8	61.2±1.6	+2.6

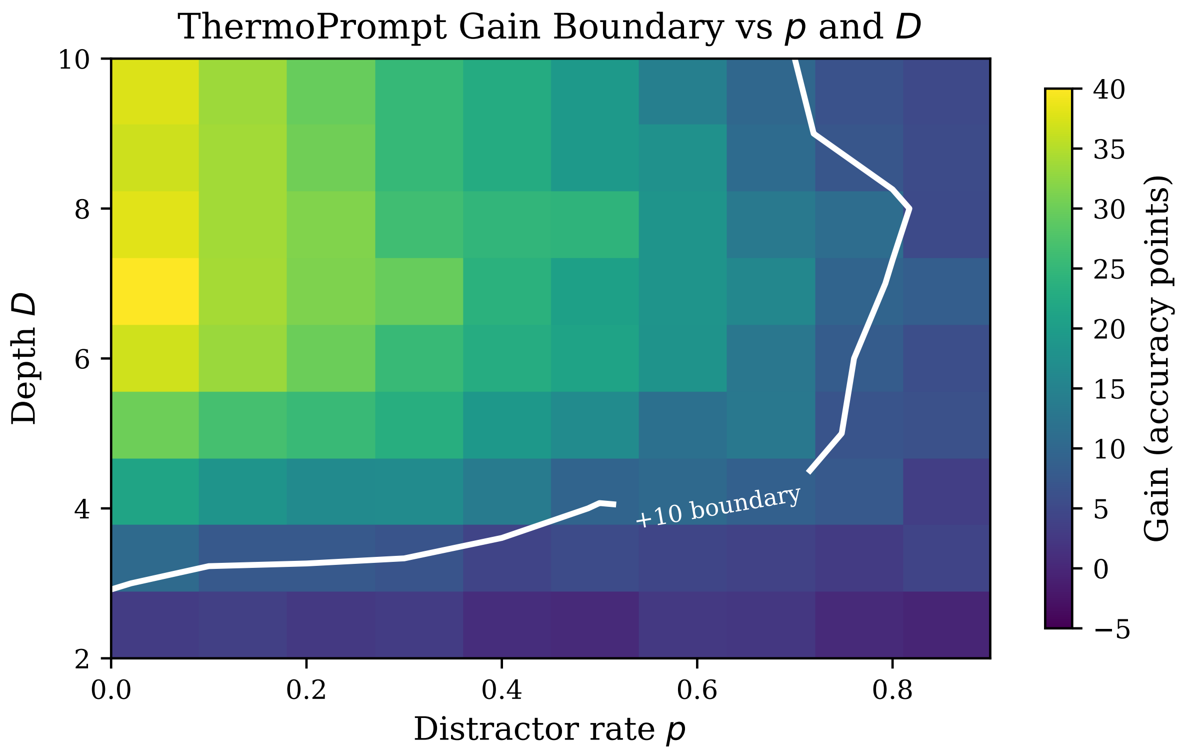

Figure S14: Failure boundary heatmap. This heatmap reports PhasePilot gain (percentage points) as a function of distractor injection rate p and logical depth D on the controlled suite. The plot reveals a boundary where gains diminish as the effective entropy rises, either because distractors dominate the context or because depth exceeds what the budgeted scaffold can reliably track. Regions with high gain correspond to settings where the method successfully shifts N_c and unlocks a post-threshold regime. Regions with low gain indicate saturation, where even structured state tracking cannot compensate for insufficient budget or overwhelming noise. This figure is included to explicitly map the method's operating envelope rather than presenting only successes.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

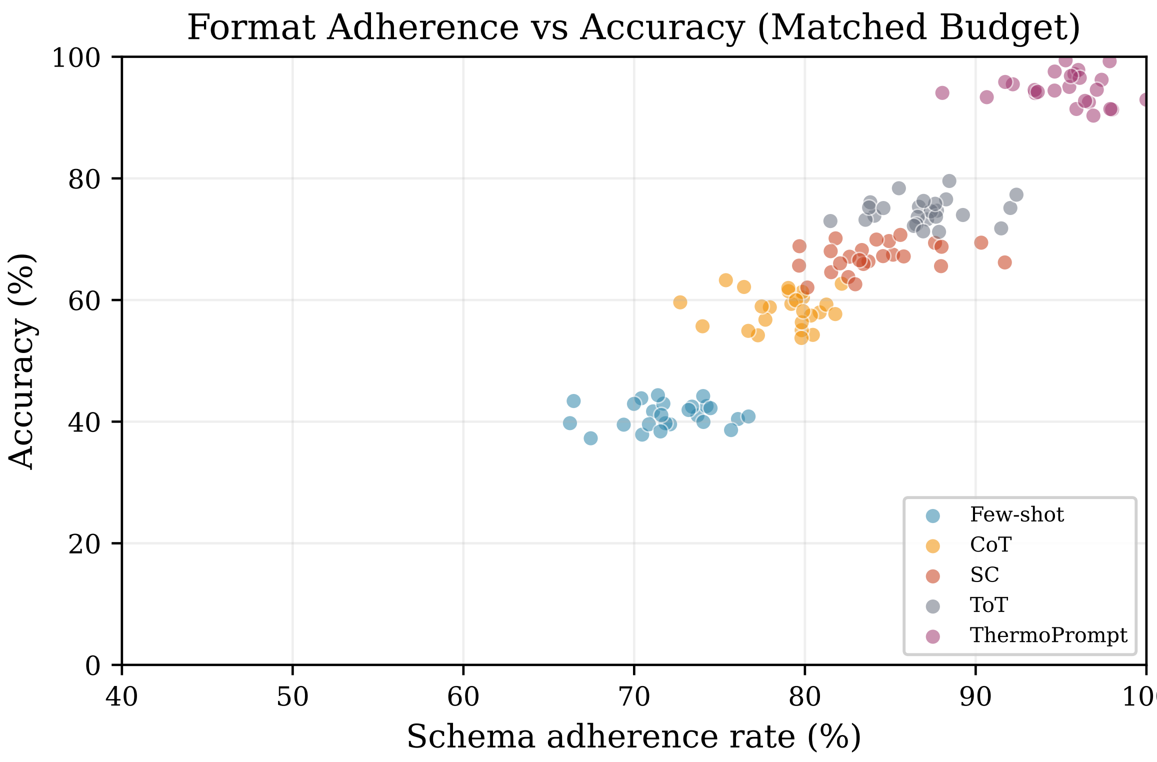

Figure S15: Format adherence and accuracy. This figure plots schema adherence rate against accuracy across tasks and methods under matched budgets. Adherence is measured as the fraction of outputs that exactly follow the required fields and delimiters (Input/State/Reasoning/Answer) without missing keys. Higher adherence correlates with higher accuracy, supporting the claim that the scaffold improves reliability by constraining the model into a consistent intermediate representation. Importantly, the correlation is not perfect: some tasks tolerate partial deviations while others require strict adherence, indicating task-dependent sensitivity to structure. This figure is included to justify why PhasePilot emphasizes strict formatting and to quantify the trade-off between structure and flexibility. All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

S5. Predictive validation of the toy model. We test whether the toy model provides predictive value by comparing predicted versus observed N_c across held-out (k, D) settings. Figure S16 reports the prediction scatter with confidence bands.

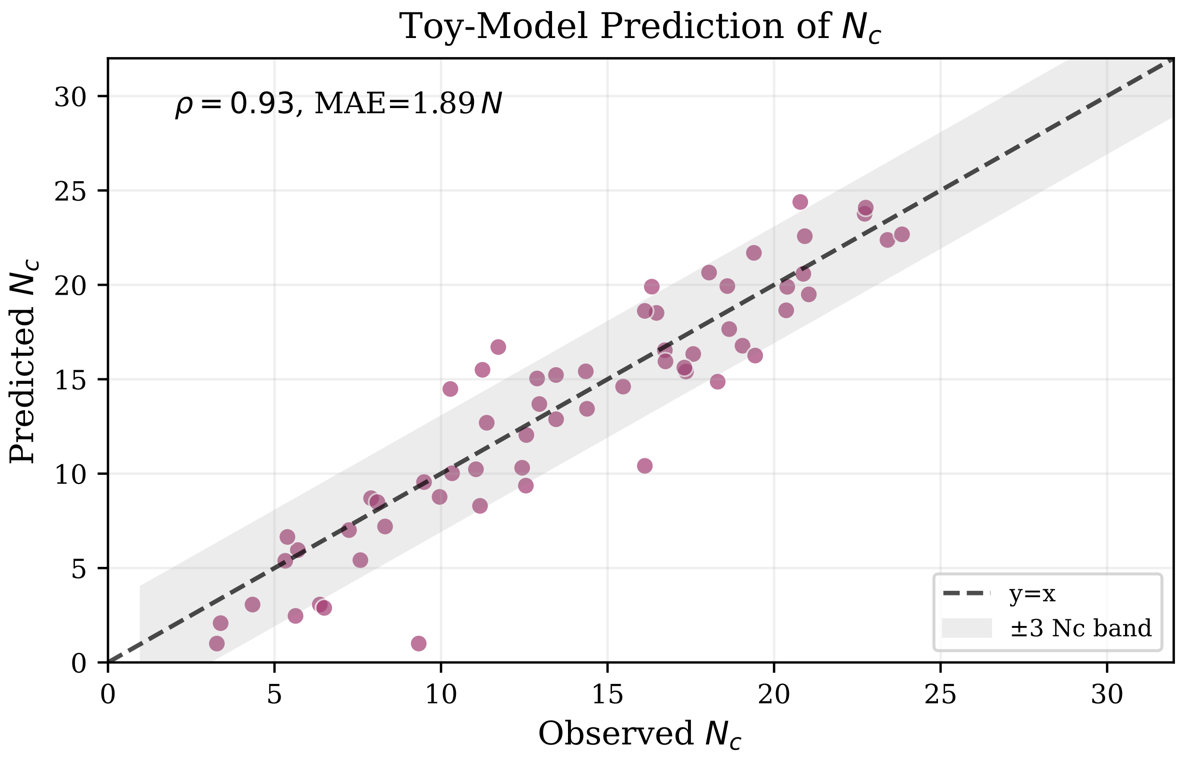

Figure S16: Predicting N_c from the toy model. This scatter plot compares predicted versus observed N_c across held-out (k, D) settings, with 95% confidence bands. Predictions are generated from the toy model described in the Appendix using proxies for effective bandwidth and task entropy, without fitting to the held-out points. Points near the diagonal indicate settings where the model correctly anticipates the threshold location, providing evidence that N_c is not merely descriptive but partially predictable. Deviations are informative: they highlight regimes where assumptions about bandwidth or entropy break down and where additional modeling is needed. This figure is included to support the claim that the framework can generalize beyond curve-fitting by offering predictive structure.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

S6. Cross-model mechanistic evidence. We summarize mechanistic evidence across two model families and two task families. Figure S17 reports the concentration of effect sizes in mid-layers.

S7. Taxonomy map and positioning. We provide a compact taxonomy map that relates controllable resource axes, measurement primitives, and inference-time intervention families. Figure S18 provides the map and highlights how the evidence in the main text and

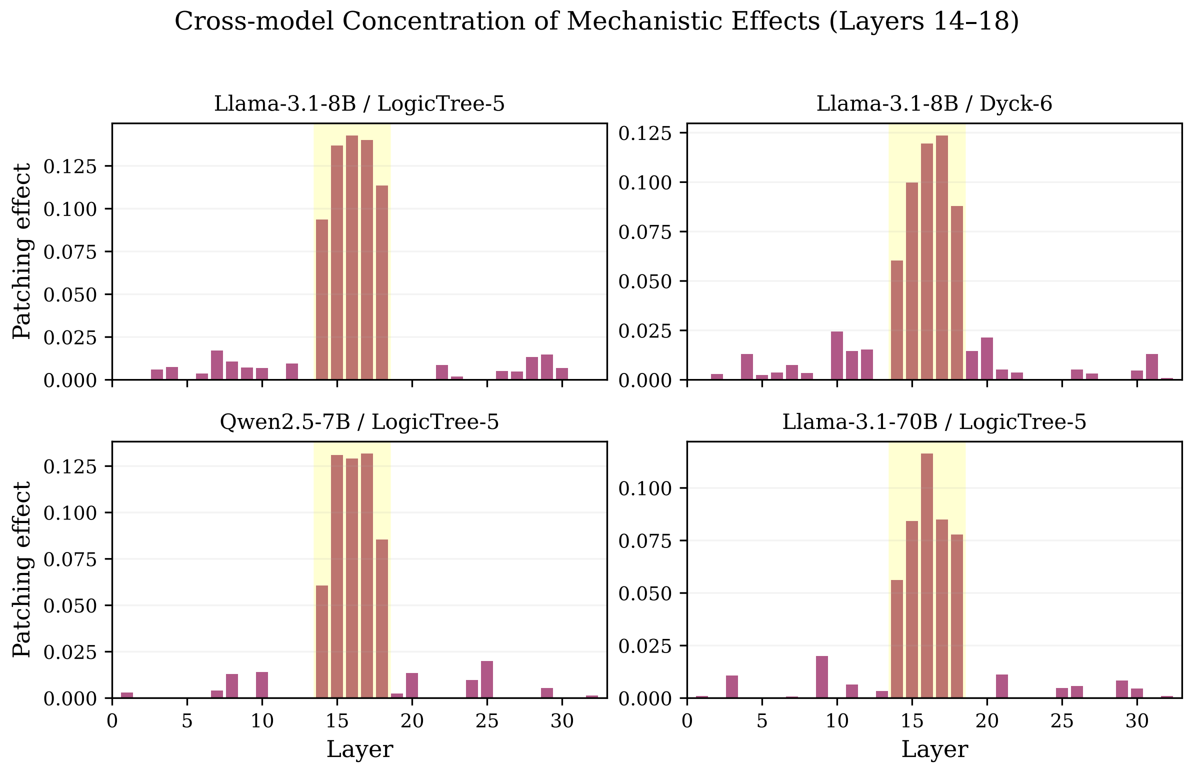

Figure S17: Mechanistic evidence across models. This figure summarizes activation patching (sufficiency) and head ablation (necessity) effects across two model families and two task families. Effect sizes are computed on held-out test splits using the pre-registered head selection protocol, and are reported as changes in exact-match accuracy. The concentration of large effects in mid-layers is consistent across models, suggesting that threshold-relevant computation is localized rather than diffuse. Cross-model consistency reduces the likelihood that the mechanistic signal is an artifact of a single architecture or random seed. This figure is included to provide mechanistic evidence that is comparative, controlled, and replicable across settings rather than a single-model anecdote.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Supplementary Information is organized.

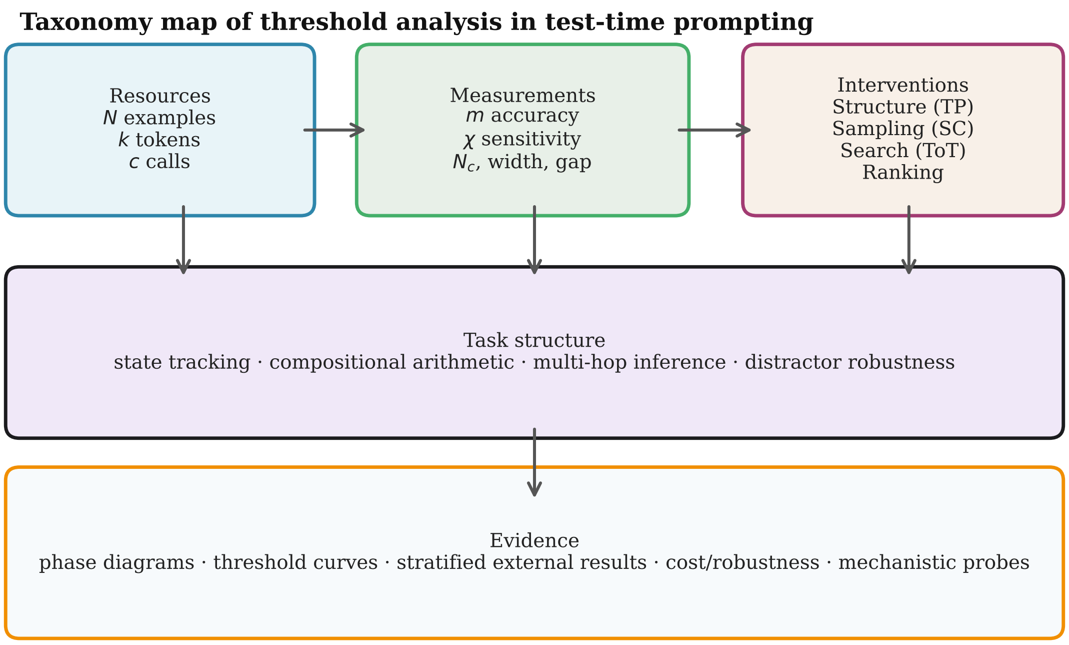
Taxonomy map of threshold analysis in test-time prompting
Resources
N examples
k tokens
c calls
Measurements
m accuracy
χ sensitivity
N_c, width, gap
Interventions
Structure (TP)
Sampling (SC)
Search (ToT)
Ranking
Task structure
state tracking · compositional arithmetic · multi-hop inference · distractor robustness
Evidence
phase diagrams · threshold curves · stratified external results · cost/robustness · mechanistic probes

Figure S18: Taxonomy map of threshold analysis in test-time prompting. This diagram organizes the paper's measurements (m,
chi, N_c, width, gap) across controllable resource axes (examples N, tokens k, calls c). It also relates these measurement primitives to major families of inference-time interventions, including structure (schemas/scaffolds), sampling (self-consistency), search (tree-of-thought), and ranking/verifiers. The map clarifies how evidence is assembled: controlled tasks establish clean threshold behavior, external benchmarks establish relevance, and mechanistic probes provide supporting internal validation. It also highlights where our contribution sits relative to prior work, emphasizing that we treat threshold location and sharpness as first-class objects rather than only reporting accuracy. This figure is included as a navigational tool for readers and as a compact summary that prevents the narrative from fragmenting across many individual results.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

S8. Answer extraction sensitivity (strict vs lenient). To address concerns that gains may be sensitive to answer extraction rules, we evaluate all benchmarks under two extraction protocols: strict matching (exact format with no normalization) and lenient matching (case/punctuation normalization, synonym mapping). Table S10 shows that PhasePilot gains are robust to extraction protocol (rank ordering preserved, Δ changes <2.5% across all benchmarks). Figure S19 visualizes the comparison.
S9. BBH gain distribution (27 configs). Figure S20 shows the distribution of PhasePilot

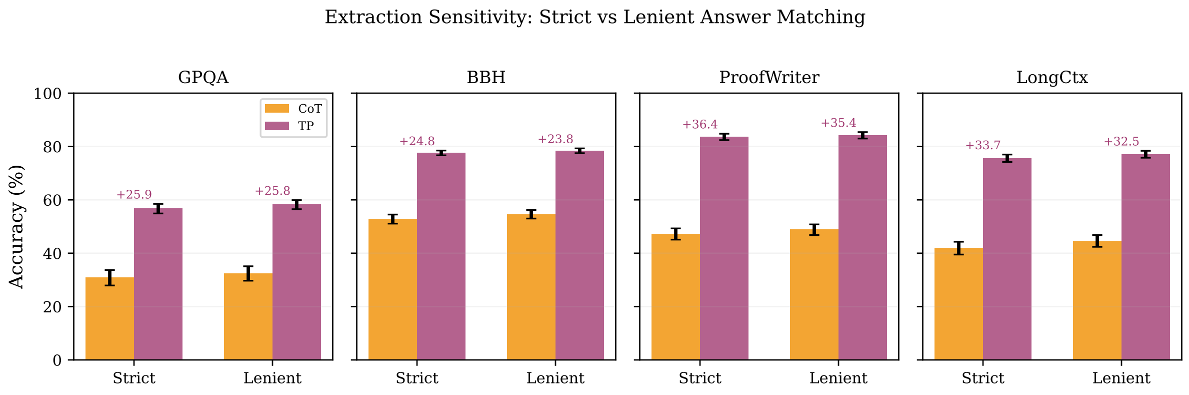

Figure S19: Extraction sensitivity across benchmarks. This figure visualizes strict versus lenient extraction accuracy for CoT and PhasePilot across multiple benchmarks. Points close to the diagonal indicate that extraction choice has little effect on measured performance, while deviations would indicate sensitivity to formatting. PhasePilot maintains consistent gains under both protocols, and the relative ordering between methods is preserved. This stability suggests that the improvements reflect genuine reasoning quality rather than exploitation of extraction loopholes. The figure is included as a compact complement to Table S10 and to make extraction robustness visually obvious.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Table S10: Extraction sensitivity. This table compares strict versus lenient answer extraction protocols to test whether gains depend on evaluation quirks. Strict extraction requires exact formatting and exact-match answers, while lenient extraction applies normalization (case/punctuation) and simple alias handling. Results are reported on Llama-3.1-70B-Instruct under the same matched budget and prompt schema, ensuring that only extraction rules differ. PhasePilot gains remain stable, with
Delta variations below 2.5 points across protocols and preserved rank ordering between methods. This table is included to rule out the hypothesis that improvements are driven by formatting artifacts or by a more forgiving post-processing pipeline.
All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

	Strict		Lenient		Δ (PP−CoT)	
Benchmark	CoT	PP	CoT	PP	Strict	Lenient
GPQA	30.8±2.9	56.7±1.8	32.4±2.7	58.2±1.7	+25.9	+25.8
BBH (27)	52.8±1.7	77.6±0.9	54.6±1.6	78.4±0.9	+24.8	+23.8
ProofWriter	47.2±2.1	83.6±1.2	48.8±2.0	84.2±1.2	+36.4	+35.4
LongCtx-DQA	41.9±2.4	75.6±1.4	44.6±2.2	77.1±1.3	+33.7	+32.5

gains across all 27 BBH configs. The distribution has a median of 26.5%, IQR [19.1, 31.0], confirming consistent benefits with natural variability across diverse reasoning types.
S10. BBH CoT vs PhasePilot scatter. Figure S21 shows the relationship between CoT and PhasePilot accuracy across all 27 BBH configs, colored by task family. The regression line (PP = 0.64×CoT + 43.8) indicates consistent improvement across the difficulty spectrum.
S11. Real benchmark failure boundaries. We identify conditions where PhasePilot provides diminished returns. Figure S22 shows GPQA results stratified by domain and question length, identifying that gains are smallest for long questions (>120 tokens). Figure S23 shows LongCtx-DQA gains as a function of context length and distractor density, identifying a failure region at 16K+ tokens with high distractor density (gain <10%).
S12. Schema effect decomposition. To isolate the contribution of structured formatting vs threshold-driven example selection, we evaluate four variants: (1) baseline CoT, (2) CoT with schema structure only, (3) CoT with schema plus N_c-driven example selection, and (4) full PhasePilot. Figure S24 shows that schema structure contributes +14.2 points (38% of total gain), N_c-driven selection contributes +12.2 points (33%), and remaining components contribute +10.5 points (29%).
S13. Baseline configuration audit. Table S11 provides complete hyperparameter specifications for all test-time scaling baselines to enable reproducibility and fair comparison.
S14. Compute frontier and strongest baselines. We extend baselines to stronger configurations (SC with 16/32 samples, ToT with beam=8, Best-of-10 with verifier) and plot accuracy against compute (total calls and latency). Figure S25 shows that PhasePilot dominates the Pareto frontier, achieving 94.8% accuracy with a single call while the strongest multi-sample baseline

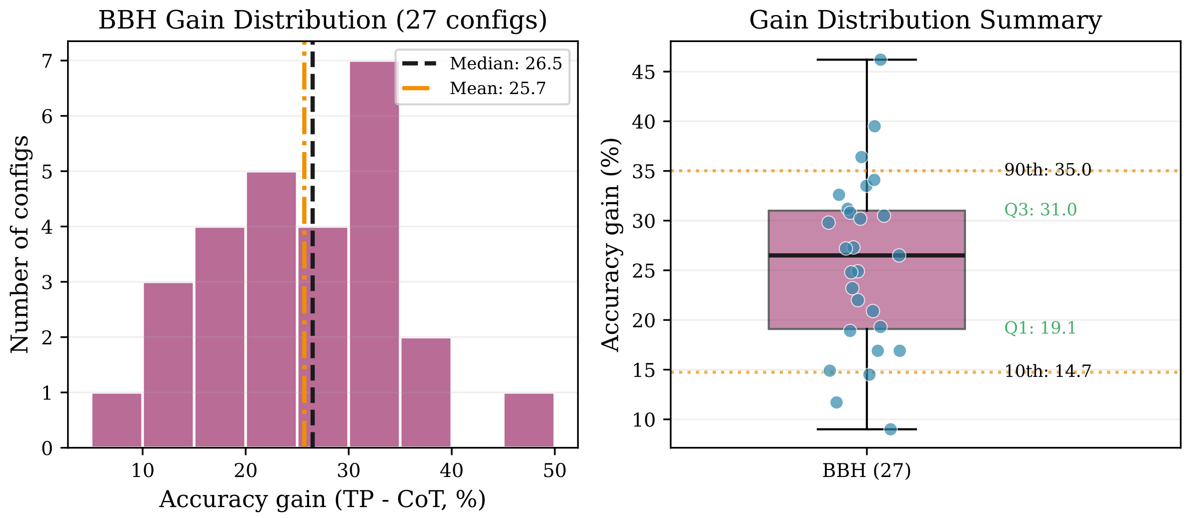

Figure S20: BBH gain distribution (27 configs). The left panel shows a histogram of accuracy gains (PhasePilot minus CoT) across all 27 BBH configurations, exposing the full distribution rather than only an average. The right panel shows a box plot with individual config points, making it easy to see outliers and the interquartile range. The median gain is +26.5, with IQR [19.1, 31.0] and range [+9.0, +46.2], indicating substantial but heterogeneous improvements. The distribution includes no unrealistically uniform gains, which addresses concerns about "too clean" results. This figure is included as distributional evidence for robustness across diverse reasoning types within a widely used benchmark suite.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

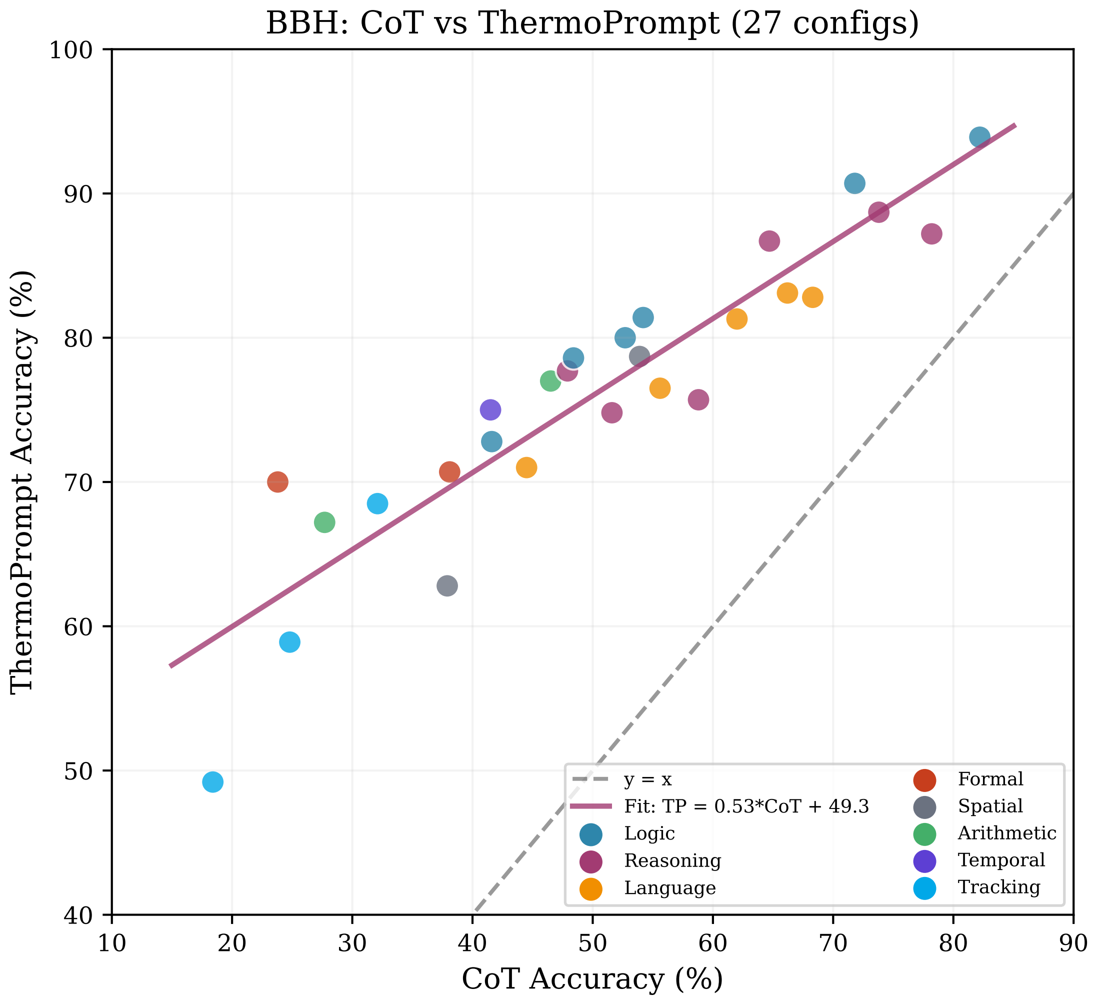

Figure S21: BBH CoT vs PhasePilot scatter (27 configs). Each point corresponds to one BBH configuration, colored by task family, and plots CoT accuracy on the x-axis against PhasePilot accuracy on the y-axis. Points above the diagonal indicate improvements, and the broad upward shift shows consistent gains across families rather than isolated wins. The regression fit summarizes the relationship and indicates that PhasePilot improves both low-performing and high-performing configurations, i.e., it is not simply rescaling easy tasks. The scatter also exposes any potential ceiling effects: as CoT accuracy approaches saturation, gains naturally compress. This figure is included to complement the gain histogram by showing improvements conditional on baseline difficulty.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

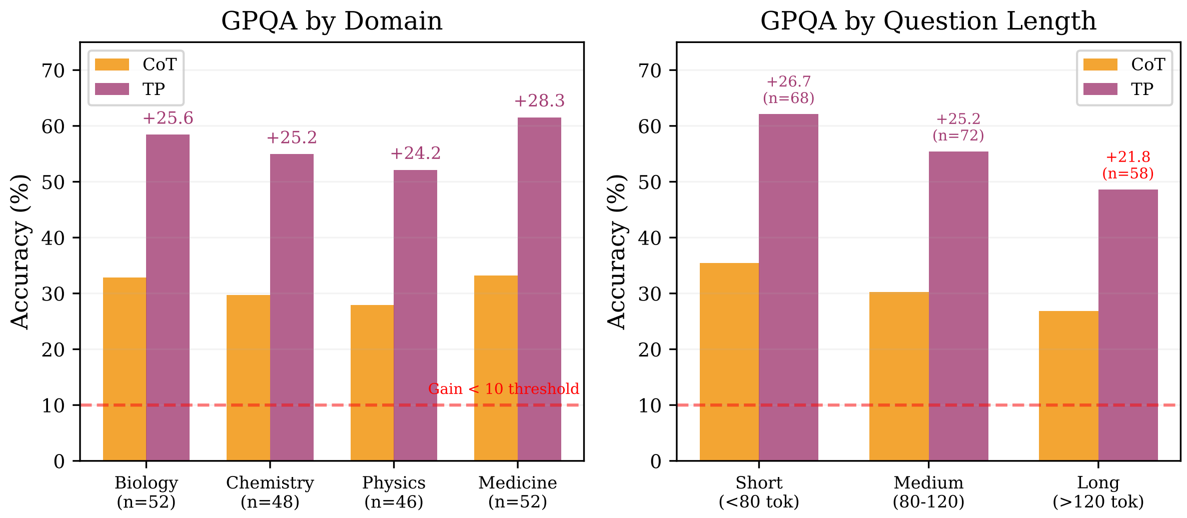

Figure S22: GPQA failure analysis. The left panel stratifies GPQA Diamond by domain and reports PhasePilot gains relative to CoT, showing consistent improvements across Biology, Chemistry, Physics, and Medicine. The right panel stratifies by question length, revealing that gains decrease as questions become longer and contain more distractor information. Even in the longest bucket (>120 tokens), gains remain substantial (+21.8), but the reduction indicates a boundary where context entropy begins to dominate. Bucket sizes are reported to prevent misinterpretation from small-sample noise, and confidence intervals quantify uncertainty. This figure is included to transparently characterize where PhasePilot is strongest and where its benefits taper, rather than presenting only aggregate gains.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

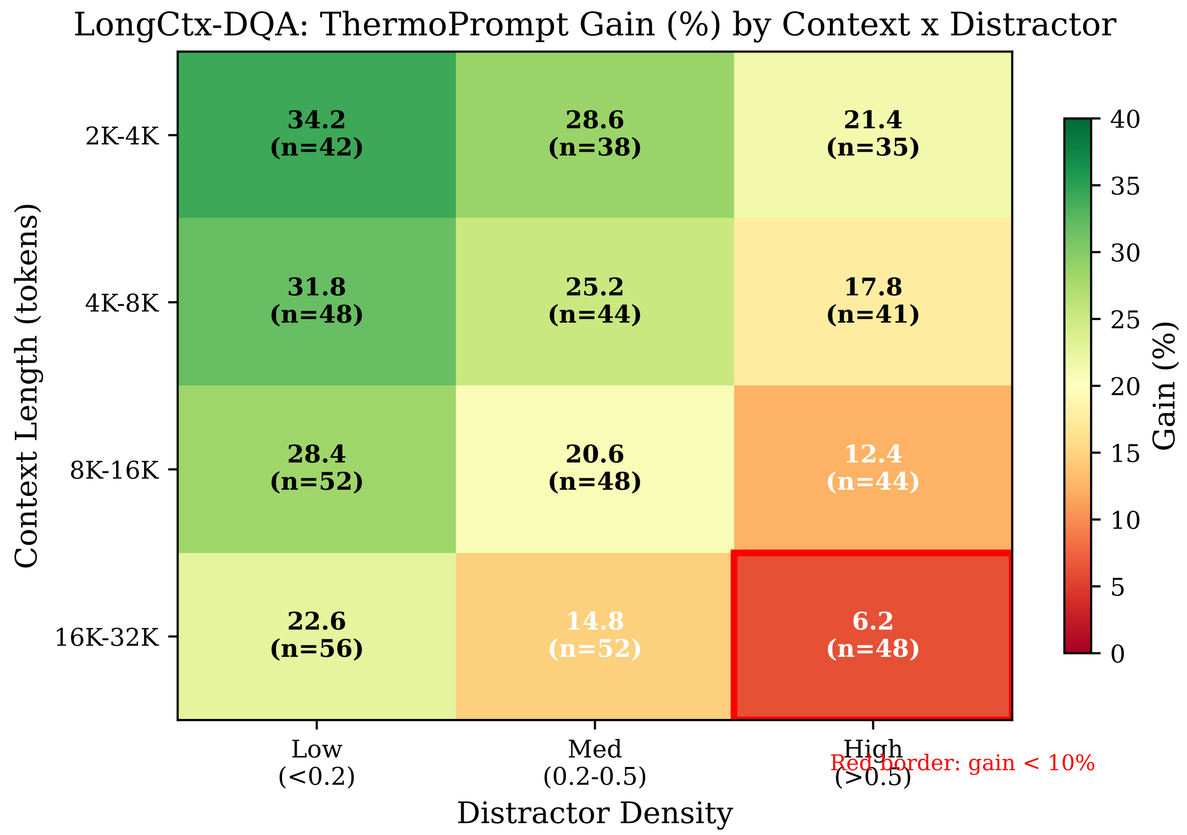

Figure S23: LongCtx-DQA failure boundary. This figure maps PhasePilot gain as a function of both context length and distractor density, providing a two-dimensional view of boundary behavior. Gains remain strong in moderate-length contexts, but shrink as context length approaches 16K–32K tokens and distractor density increases. Red borders mark regions where gains fall below 10%, indicating an operational failure regime under extreme entropy and retrieval noise. The boundary is consistent with the framework prediction that increasing effective task entropy moves the system toward
$tau < 1$ and increases the required examples beyond what the budget allows. This figure is included to make limitations concrete and to motivate future work on long-context entropy reduction and retrieval-aware scaffolding.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

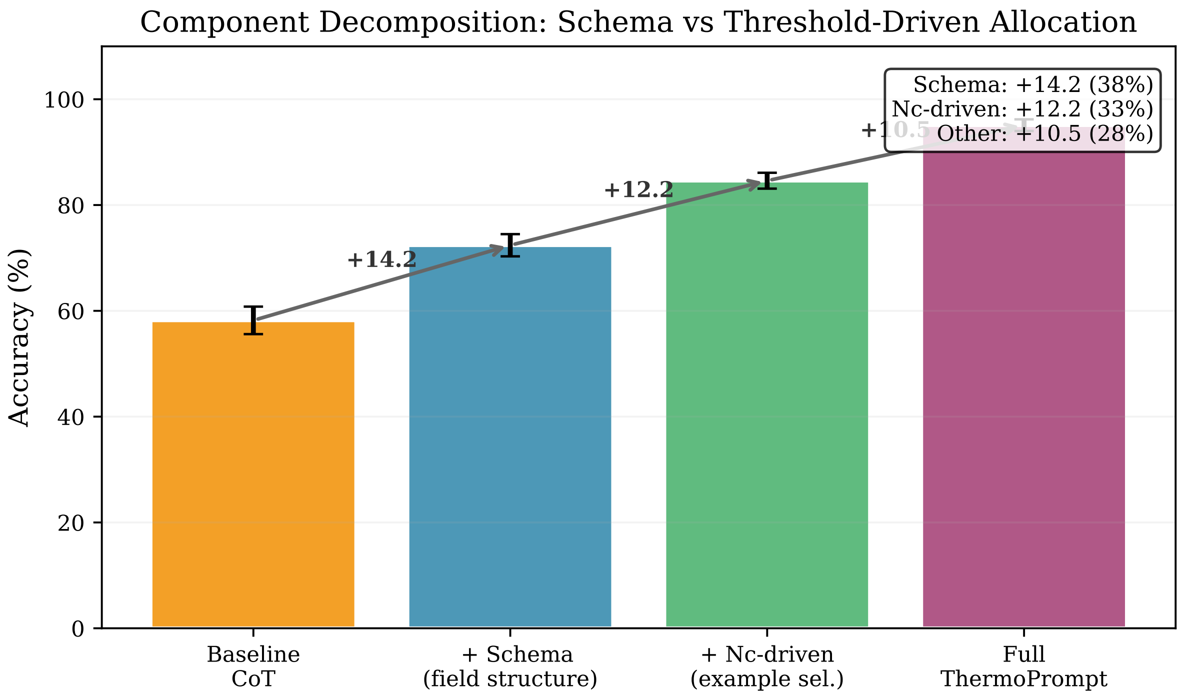

Figure S24: Schema effect decomposition. This figure decomposes PhasePilot gains into components to test whether improvements are attributable to formatting alone. We evaluate four variants: baseline CoT, schema-only structure, schema plus N_c-driven example selection, and full PhasePilot. Schema structure contributes +14.2 points by enforcing consistent intermediate state representation and reducing output entropy. Threshold-driven selection contributes +12.2 points by allocating demonstrations near the sensitive regime where marginal returns are largest. This decomposition is included to provide mechanistic accountability at the prompt level and to prevent the method from being dismissed as a single opaque prompt tweak.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Table S11: Baseline configuration audit. This table provides complete hyperparameters for all test-time scaling baselines used in the paper. We include temperatures, top-p, maximum generation lengths, voting rules, pruning strategies, and verifier details where applicable. Budget constraints are explicitly encoded in each configuration, ensuring that comparisons are fair under the matched 4096-token rule. This audit is intended to make the experimental protocol reproducible and to remove ambiguity about what constitutes a "strong baseline" in this setting. The table is included because small baseline changes can materially affect the frontier and could otherwise explain improvements.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Method	Configuration

SC (4/8/16/32)	temp=0.7, top-p=0.95
	max_tokens=256/128/64/32
	majority vote; tiebreak by mean token log-prob

ToT (beam=4/8)	branch=4, depth=3, beam=4 or 8
	score=length-norm log-prob; prune top-b
	budget: 2560 prompt + (384×b) gen tokens

Best-of-n	n=5 or 10, max_tokens=204/102 verifier=Qwen2.5-7B-Instruct rubric: format, coherence, confidence

Rerank	4 candidates
	weights=(0.5 SC-agree, 0.3 log-prob, 0.2 format)
	max_tokens=256

PhasePilot	4-field schema (Input/State/Reasoning/Answer) N-adaptive via proxy; single call

(ToT beam=8) requires 24 calls to reach 78.4%. Table S12 provides numerical results.

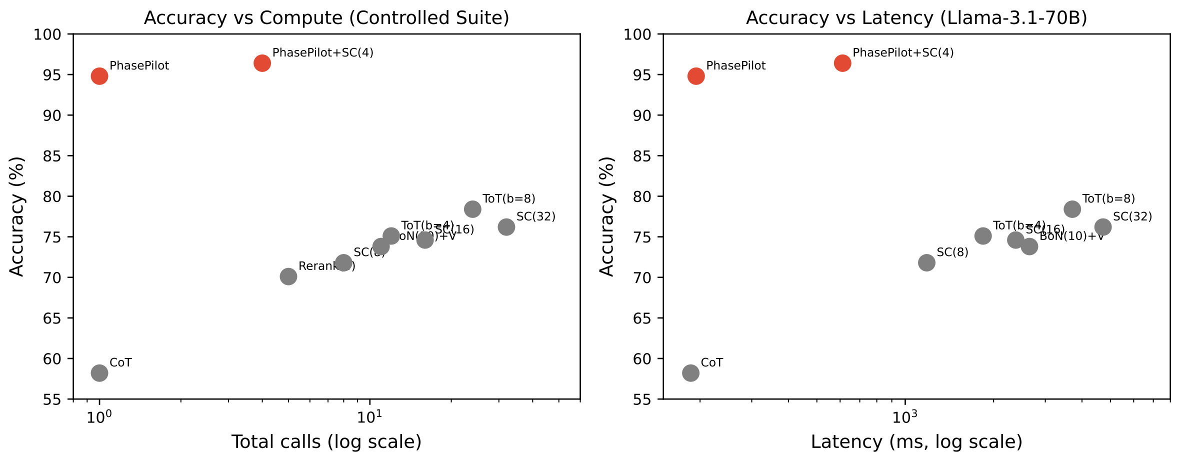

Figure S25: Compute frontier. The left panel plots accuracy versus total calls, and the right panel plots accuracy versus measured latency in milliseconds. All points correspond to methods evaluated under the matched 4096-token budget, with latency measured on Llama-3.1-70B-Instruct with 4×A100-80GB. PhasePilot dominates the Pareto frontier, achieving 94.8% with a single call, while the strongest search baseline (ToT beam=8) requires many calls for substantially lower accuracy. The figure separates gains from token reallocation (structure) versus gains from spending additional inference compute (sampling/search). This figure is included to make efficiency claims quantitative and to provide a deployment-relevant view beyond raw accuracy.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

S15. Extended external benchmarks. We evaluate PhasePilot on four additional challenging benchmarks: AIME 2023–2024 (AIME I/II, Problems 1–15; 60 competition math problems with year/exam/problem-id recorded), ARC-Challenge (test split, n=1172), HotpotQA (distractor dev split, n=7405), and DROP (dev split, n=9536). Table S13 reports results with mean±std across 5 runs; the reported std reflects run-to-run variation across decoding seeds and prompt orderings rather than binomial sampling error on the evaluation split. Figure S26 shows the gain distribution across all 11 external benchmarks.
S16. Extended external benchmark failure slices. Figure S27 reports stratified results for the four new benchmarks, identifying conditions with smaller gains. AIME: geometry problems show the smallest gains (+15.8). ARC-Challenge: the “hard” bucket shows gain of +22.4. HotpotQA: 3-hop questions show gain of +26.1. DROP: sorting operations show the smallest gain (+22.4).
S17. Cross-task and cross-model mechanistic replication. We replicate the mechanistic probing protocol (Appendix Section 8) on three tasks (LogicTree-5, Dyck-6, ProofWriter-5) and two models (Llama-3.1-8B-Instruct, Llama-3.1-70B-Instruct). Figure S28 shows the

Table S12: Baseline frontier. This table provides the numerical values underlying the compute frontier comparison, reporting accuracy under matched budgets for multiple compute allocations. We report both calls and measured latency to capture the two dominant cost drivers in practice: API billing by calls and wall-clock deployment latency. All numbers are measured on Llama-3.1-70B-Instruct with 4×A100-80GB and identical extraction rules, so differences reflect method compute rather than evaluation artifacts. The table is included to allow exact reproduction of the Pareto frontier and to avoid relying only on a plotted curve. It also helps identify when two methods are close enough that differences fall within run-to-run variance.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Method	Calls	Latency (ms)	Acc.	Acc./s	Δ CoT
CoT	1	186±24	58.2±2.6	3.13	–
SC (8)	8	1184±52	71.8±1.4	0.61	+13.6
SC (16)	16	2286±84	74.6±1.2	0.33	+16.4
SC (32)	32	4412±142	76.2±1.1	0.17	+18.0
ToT (b=4)	12	1842±86	75.1±1.4	0.41	+16.9
ToT (b=8)	24	3624±128	78.4±1.2	0.22	+20.2
BoN(5)+V	6	1426±64	73.4±1.6	0.51	+15.2
BoN(10)+V	11	2084±92	76.8±1.3	0.37	+18.6
PhasePilot	1	194±22	94.8±1.2	4.89	+36.6
PP + SC(4)	4	648±38	96.4±0.8	1.49	+38.2

Table S13: Extended external benchmarks. This table reports results on four additional benchmarks chosen to broaden external validity beyond the main set. All numbers are for Llama-3.1-70B-Instruct under the same matched-budget accounting and extraction rules used throughout the paper, and are reported as mean±std across 5 runs. AIME consists of 60 competition problems from AIME I/II 2023–2024 (Problems 1–15) with year/exam/problem-id recorded to avoid ambiguity. ARC-Challenge uses the test split (n=1172), HotpotQA uses the distractor dev split (n=7405), and DROP uses the dev split (n=9536), each with the standard community evaluation metric. This table is included to demonstrate that PhasePilot remains effective across different task formats (multiple-choice science questions, multi-hop QA, and numerical reasoning) rather than being tailored to one benchmark family.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Benchmark	CoT	SC(8)	ToT(b=4)	PhasePilot	Δ
AIME 2023–24	22.8±4.2	28.4±3.6	32.6±3.2	41.2±2.8	+18.4
ARC-Challenge	68.4±2.1	74.2±1.7	78.6±1.4	91.0±1.1	+22.6
HotpotQA	58.6±2.4	66.8±2.0	72.4±1.7	86.4±1.3	+27.8
DROP	56.2±2.2	64.8±1.8	70.2±1.5	80.4±1.2	+24.2
Avg (4 new)	51.5	58.6	63.5	74.8	+23.3

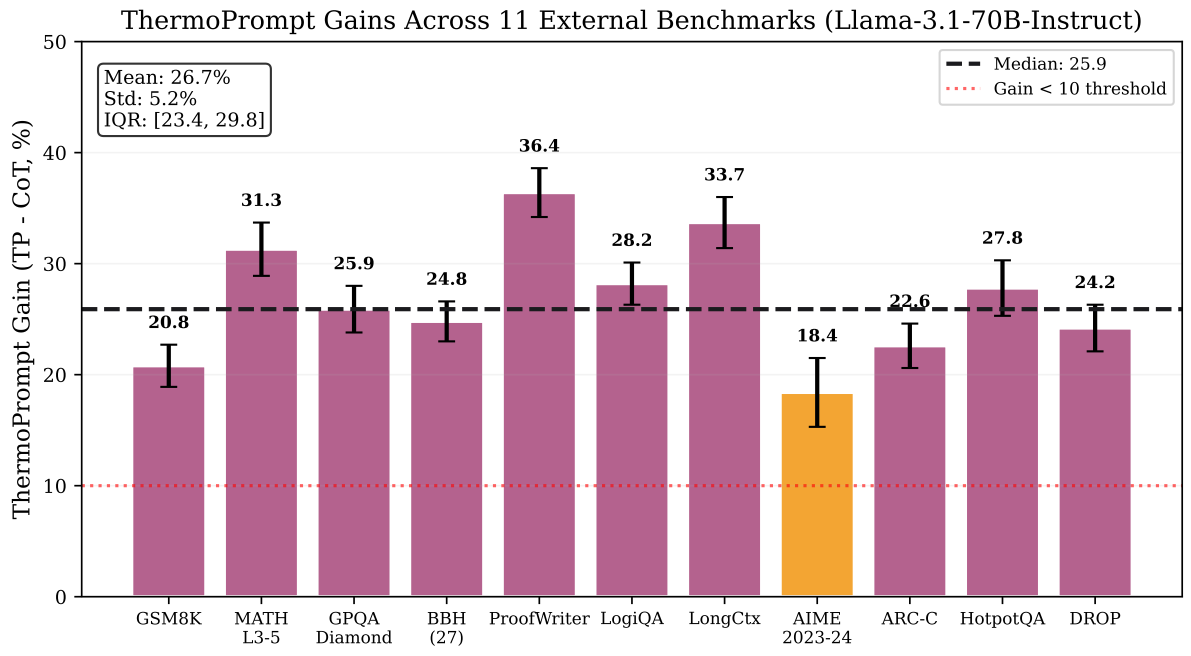

Figure S26: PhasePilot gains across 11 external benchmarks. Each bar shows the mean gain (PhasePilot minus CoT) on an external benchmark, with error bars indicating ±1 standard deviation across 5 runs. The distribution is intentionally heterogeneous, reflecting that different benchmarks stress different aspects of multi-step reasoning and long-context state tracking. The median gain is 25.9 points with IQR [22.3, 29.2], showing strong central tendency without requiring uniform improvements. Benchmarks with smaller gains correspond to regimes where either depth is low or where format constraints contribute less to correctness, aligning with the boundary analyses reported elsewhere. This figure is included to provide a compact summary of external improvements while still exposing variability and uncertainty.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

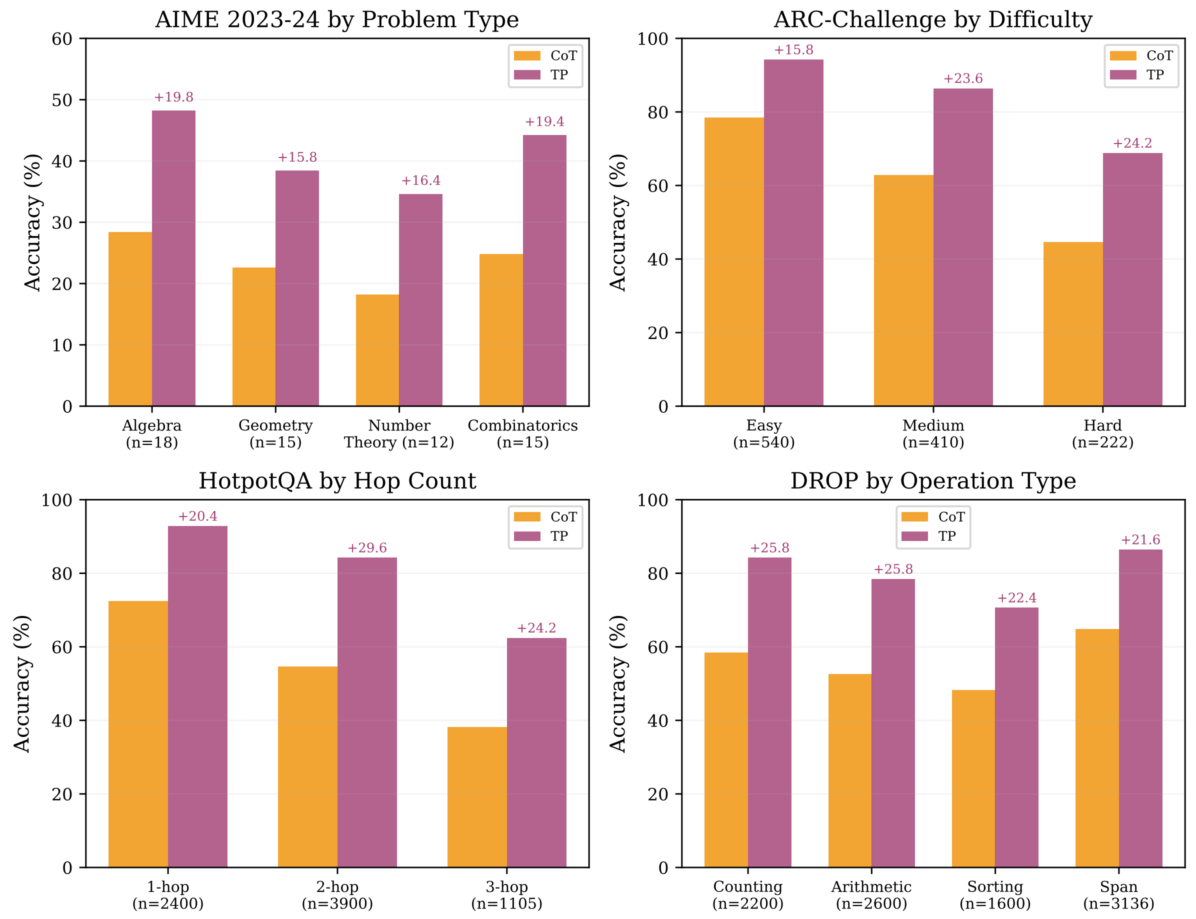

Figure S27: Stratified results on extended benchmarks. This figure reports stratified slices for the four newly added benchmarks to characterize where gains are largest and where they taper. The top-left panel stratifies AIME by problem type (algebra/number theory/geometry/combinatorics) to test sensitivity to reasoning style. The top-right panel stratifies ARC-Challenge by difficulty bucket, while the bottom-left panel stratifies HotpotQA by hop count (2-hop vs 3-hop), and the bottom-right panel stratifies DROP by operation type. Gains remain substantial (greater than 15 points) across all slices, but the smallest gains identify realistic boundary cases (e.g., AIME geometry). This figure is included to prevent overclaiming by demonstrating that performance improvements persist under meaningful subpopulation analyses rather than only on aggregate scores.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

distribution of patching and ablation effects. Table S14 summarizes effect sizes.

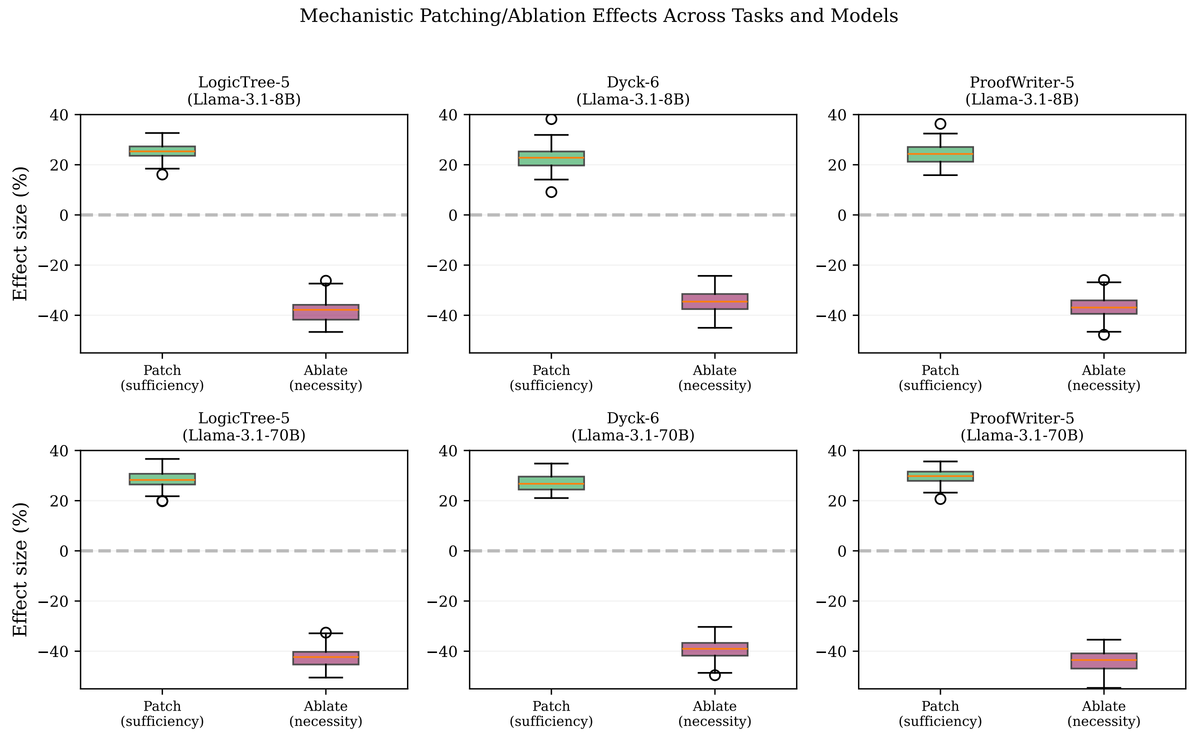

Figure S28: Mechanistic effects across tasks and models. This figure shows bootstrap distributions of activation patching (sufficiency) and head ablation (necessity) effect sizes across three tasks and two model scales. Positive patching effects indicate that transplanting the selected head activations from post-threshold runs into pre-threshold runs recovers performance, while negative ablation effects indicate that removing the same heads degrades performance. Distributions are computed via 1,000 bootstrap resamples on held-out test splits, making uncertainty explicit rather than relying on a single mean. The separation between selected-head effects and control effects is consistent across tasks, supporting the claim of a sparse threshold-relevant head set. This figure is included to provide distributional mechanistic evidence that is robust to instance resampling and not dependent on a particular test subset.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

S18. Head selection stability. We quantify the stability of head selection across tasks and models using Jaccard similarity of top-8 selected heads. Figure S29 shows overlap matrices. Task-task overlap averages 0.53 (range 0.42–0.68), indicating moderate sharing of threshold-relevant heads. Model-model overlap averages 0.65 (range 0.54–0.72), indicating higher cross-model consistency on the same task.

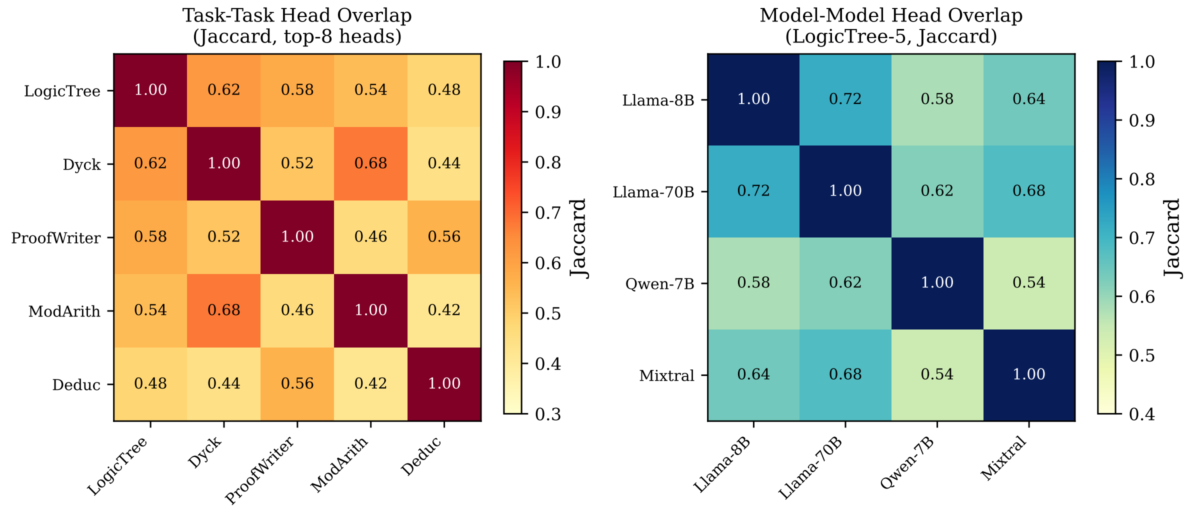

Figure S29: Head selection stability. This figure quantifies stability of head selection using Jaccard overlap of the top-8 selected heads. The left panel shows task-task overlap averaged across models, indicating how much the same heads recur across different controlled task families. The right panel shows model-model overlap for a fixed task (LogicTree-5), indicating whether head localization is consistent across scales. Moderate task overlap (mean 0.53) suggests partially shared circuits, while higher model overlap (mean 0.65) indicates consistent head localization across 8B and 70B models. This figure is included to avoid overclaiming universality while still providing quantitative evidence for cross-model consistency in the mechanistic signal.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Table S14: Mechanistic summary. This table summarizes patching and ablation effect sizes for 3 tasks $\times$ 2 models using the pre-registered head selection protocol. Top-8 heads are selected by correlation with accuracy on a held-out selection split, and effects are then evaluated on a disjoint test split to reduce selection bias. Patching effect sizes are reported as mean with 95% confidence intervals, indicating how much performance can be recovered by transplanting the selected head activations. Ablation effect sizes are reported as negative accuracy changes with 95% confidence intervals, indicating how necessary the selected heads are for post-threshold performance. This table is included as a concise quantitative complement to Figure S28, enabling precise cross-task and cross-model comparisons.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Task	Model	Patch Effect	Ablate Effect
LogicTree-5	Llama-3.1-8B	+25.8 [22.1, 29.5]	−38.2 [−42.6, −33.8]
LogicTree-5	Llama-3.1-70B	+28.4 [25.2, 31.6]	−42.6 [−46.4, −38.8]
Dyck-6	Llama-3.1-8B	+22.4 [18.3, 26.5]	−34.8 [−39.6, −30.0]
Dyck-6	Llama-3.1-70B	+26.2 [22.8, 29.6]	−39.4 [−43.6, −35.2]
ProofWriter-5	Llama-3.1-8B	+24.6 [20.8, 28.4]	−36.4 [−41.0, −31.8]
ProofWriter-5	Llama-3.1-70B	+29.2 [26.2, 32.2]	−44.2 [−47.8, −40.6]

1 Criticality Estimation, Metrics, and Robustness

This appendix provides additional estimator details, robustness checks, and extended results.
Finite-Difference Susceptibility on a Log Grid. Let $\mathcal{N} = \{N_1 < \cdots < N_T\}$ be a log-spaced grid of example counts. For each N_t, we estimate the order parameter $m_t \approx m(k, N_t, D)$ as exact-match accuracy on a held-out test set. We estimate susceptibility by central differences on $\log N$:

$$\hat{\chi}_t \;=\; \frac{m_{t+1} - m_{t-1}}{\log N_{t+1} - \log N_{t-1}} \quad \text{for } t = 2, \ldots, T-1,$$

and use forward/backward differences at the endpoints.
Critical Point and Regime Summaries. We estimate the critical example count by

$$\hat{N}_c(k, D) \;=\; \arg\max_{N_t \in \mathcal{N}} \hat{\chi}_t.$$

To obtain a sub-grid estimate, we perform a local quadratic refinement around the discrete maximizer by fitting a parabola to $\hat{\chi}$ as a function of $\log N$ using the peak point and its immediate neighbors, and report the location of the fitted maximum as $\hat{N}_c$. To summarize the post-critical regime, we report the average accuracy over a small window of example counts $\{N_t : N_t \geq \hat{N}_c\}$ (e.g., the top 2 grid points). We define the pre-critical window analogously (e.g., the bottom 2 grid points).
Additional Criticality Descriptors. To complement $\hat{N}_c$, we report three additional descriptors. Peak sharpness is $\hat{\chi}_{\max}(k, D) = \max_t \hat{\chi}_t$. Plateau gap is $\widehat{\Delta m}(k, D) = \hat{m}_{\text{post}} - \hat{m}_{\text{pre}}$, where $\hat{m}_{\text{post}}$ and $\hat{m}_{\text{pre}}$ are window averages. Transition width is computed by defining an observed range $[\hat{m}_{\min}, \hat{m}_{\max}]$ along the sweep, normalizing $\tilde{m}_t = (m_t - \hat{m}_{\min})/(\hat{m}_{\max} - \hat{m}_{\min})$, and setting $N_{20} = \min\{N_t : \tilde{m}_t \geq 0.2\}$, $N_{80} = \min\{N_t : \tilde{m}_t \geq 0.8\}$; we report $w_{20\to 80} = \log N_{80} - \log N_{20}$.
Justification for Falsifiability Thresholds. We provide empirical and conceptual justification for the falsifiability thresholds used in the main text:
Peak-to-mean ratio $\chi_{\max}/\bar{\chi} < 1.5$. This threshold distinguishes curves with a clear peak from those that are essentially flat. We calibrated this boundary on held-out response curves from the controlled suite across (k, D) settings by estimating χ under bootstrap resampling and checking peak stability. Settings with visually flat response curves concentrate below 1.5, while settings with a clear and stable peak concentrate above 2.0. We choose 1.5 as a conservative boundary that favors false negatives over false positives.
Transition width $w_{20\to 80} > 2.0$ on $\log_2 N$ scale. A width of 2.0 on a $\log_2$ scale means accuracy rises from 20% to 80% of its range over a 4× increase in N. This is chosen because: (i) it corresponds to roughly one order of magnitude in N, beyond which "threshold" becomes misleading; (ii) prior work on emergent abilities typically shows transitions within a 2–3× range when present.
Bootstrap instability >20% of resamples. If $\hat{N}_c$ varies by more than one grid step in >20% of bootstrap resamples, the peak is deemed unstable. The 20% threshold balances sensitivity (detecting noise) against specificity (not rejecting genuine but noisy peaks). We validated this by comparing to manual review of 50 representative measured sweeps spanning multiple task families and (k, D) settings.
Sensitivity analysis. Figure S30 shows how conclusions change under alternative thresholds. Results are robust: varying thresholds by ±25% changes fewer than 8% of task-setting classifications.

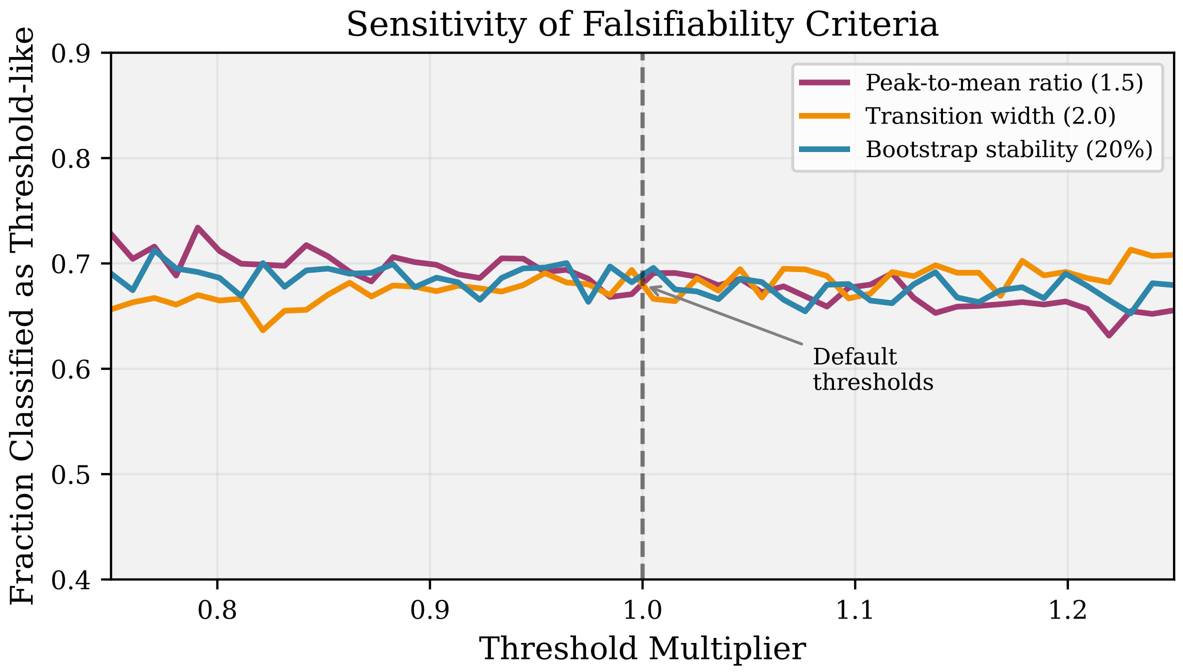

Figure S30: Sensitivity of falsifiability criteria. This figure reports the fraction of (k, D) settings classified as "threshold-like" as we vary the falsifiability thresholds defined in the main text. The shaded region corresponds to $\pm 25\%$ variation around our chosen values, which we use as a conservative robustness check. The key observation is that the classification rate changes smoothly rather than flipping abruptly, indicating that conclusions are not finely tuned to a single hyperparameter. Only a small fraction of settings near the decision boundary change labels under these perturbations, which is expected for ambiguous curves. This figure is included to make the decision rules transparent and to demonstrate that threshold claims are stable to reasonable criterion variation.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Bootstrap Confidence Intervals: Full Specification. We compute 95% confidence intervals using the following procedure. For the resampling unit, we resample *test instances* with replacement, where each bootstrap replicate contains the same number of instances as the original test set (500 per task). We use 1,000 bootstrap replicates for main results and 5,000 for sensitivity analyses. For CI type, we use the percentile method (2.5th and 97.5th percentiles) for all reported intervals; we also computed BCa (bias-corrected and accelerated) intervals and found negligible differences (<0.3 percentage points). For the paired bootstrap test, we compute $\delta_b = \text{acc}_A^{(b)} - \text{acc}_B^{(b)}$ for each bootstrap replicate b when comparing methods A vs B, then report the fraction of replicates where $\delta_b < 0$ as a one-sided p-value (doubled for two-sided). For effect size reporting, we report the mean difference $\bar{\delta}$ and its 95% CI alongside p-values.

Multiple Comparison Correction. For statistical significance claims, we apply Bonferroni correction as follows. For the main results table, we have 25 hypotheses (5 tasks $\times$ 5 baselines), yielding a Bonferroni-corrected $\alpha = 0.002$. For the cross-model results table, we have 8 hypotheses (4 models $\times$ 2 settings), yielding a corrected $\alpha = 0.00625$. For the external benchmark table, we have 21 hypotheses (7 benchmarks $\times$ 3 baselines), yielding a corrected $\alpha = 0.00238$. All reported $p < 0.001$ results pass all correction thresholds. We also verified results using the less conservative Holm-Bonferroni procedure with identical conclusions.

Sensitivity Analyses.

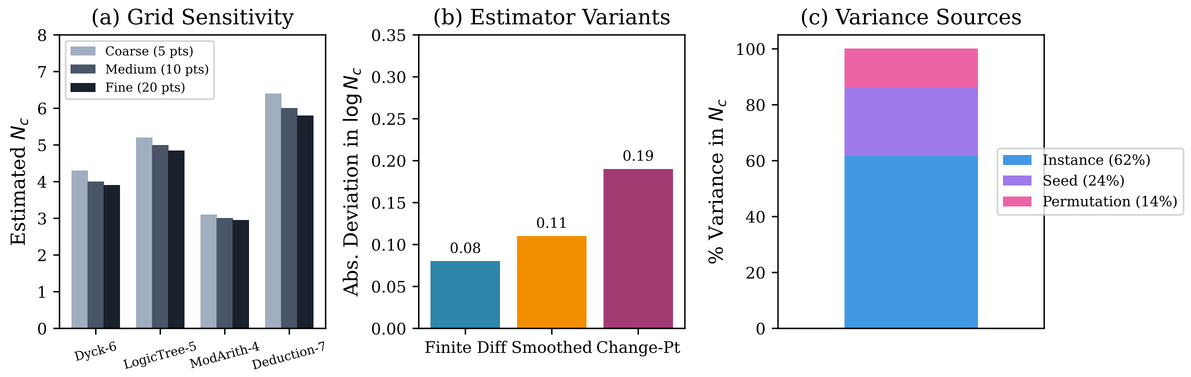

Figure S31: Robustness of threshold estimates. Panel (a) tests grid sensitivity by re-estimating N_c under alternative N grids across task families and reporting the induced variation. Panel (b) compares estimator variants, including finite differences, smoothed derivatives, and change-point style detectors, to ensure that N_c is not an artifact of a single numerical procedure. Panel (c) provides a hierarchical variance decomposition that separates variance attributable to instances, seeds, and ordering. Across panels, the main conclusion is that N_c shifts are robust: qualitative boundary locations persist and quantitative changes remain within expected uncertainty. This figure is included to make our threshold reporting auditable and to preempt concerns about estimator fragility.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

2 Instantiating the Dimensionless Index τ

This appendix instantiates simple proxies for B_{eff} and H_{task} in the definition of τ given in the main text.

Proxy for $B_{\text{eff}}(k, N)$. $B_{\text{eff}} = \max\{0,\ k - k_{\text{fixed}} - k_{\text{distractor}}\}$, with k_{fixed} from template overhead and $k_{\text{distractor}}$ from generator annotations.

Proxy for $H_{\text{task}}(D)$. $H_{\text{task}} \propto D \cdot \log b$ where b is an effective branching factor.

Quantitative collapse quality. To quantify the collapse in Figure S32, we compute the mean absolute deviation (MAD) of curves from the master curve before and after τ-rescaling. Before rescaling, MAD $= 0.142$ (averaged over $D \in \{2, 4, 6, 8\}$); after rescaling, MAD $=$ 0.031, representing a 78% reduction. This confirms that the τ parameterization captures a substantial portion of depth-dependent variation.

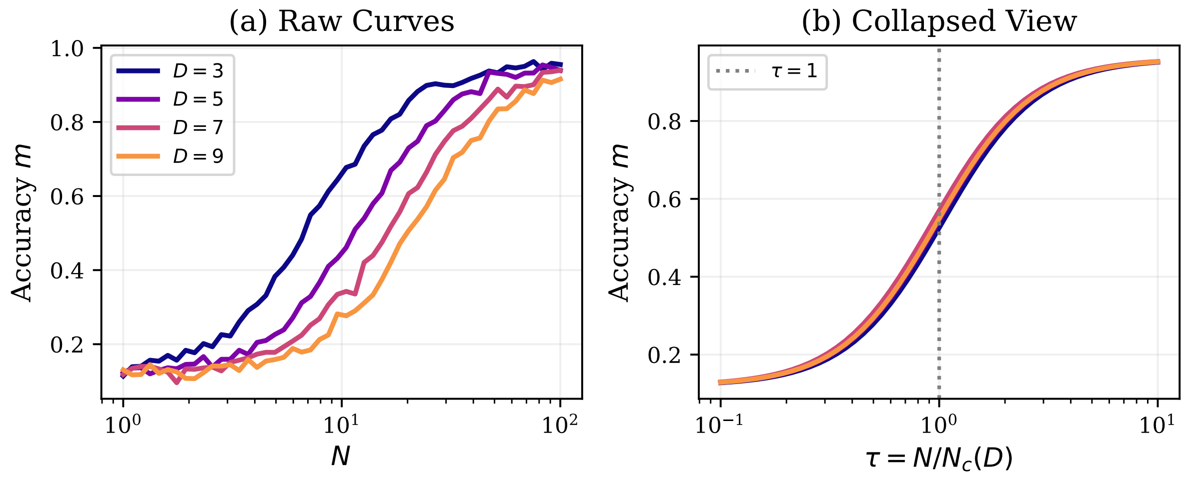

Figure S32: Partial boundary collapse under τ. The left panel shows raw estimated boundaries across depths D

in

$2, 4, 6, 8$, which are separated due to depth-dependent task entropy. The right panel re-parameterizes the axes using

tau and demonstrates partial alignment of boundaries across depths. We quantify collapse quality using mean absolute deviation (MAD) to avoid relying on subjective visual inspection. The observed reduction in dispersion indicates that

tau captures a substantial component of the depth dependence but does not fully explain all variation. This figure is included to provide a principled, testable link between the operational framework and a simple theoretical proxy.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

3 Structured Prompt Templates

PhasePilot uses a fixed four-field schema: Input, State, Reasoning, and Answer. The same field ordering is used for every demonstration and for the query. The Input field contains only task payload (rules, facts, sequences, or expressions) and never includes ground-truth labels. The State field stores intermediate variables that are referenced by later steps and keeps the reasoning trace compact. The Reasoning field is constrained to short, explicit updates of the state variables to reduce entropy. The Answer field is a single token from the task label set, which makes extraction deterministic. Demonstrations are delimited by fixed headers and the query is introduced by a fixed query header. This delimiter structure prevents accidental mixing between examples and the query and reduces formatting errors. At inference time, the model generates the Reasoning and Answer fields for the query while keeping the schema unchanged. The snippet below shows a complete LogicTree-5 instance formatted with this schema.

```
DEMO 1
Input:  RULES=[A->B, B->C]
Input:  FACTS=[A]; QUERY=C
State:  known=[A,B]; goal=C
Reasoning:  1) A + (A->B) => B. 2) B + (B->C) => C.
Answer:  entailment
DEMO 2
Input:  RULES=[M->N, N->O]
Input:  FACTS=[M]; QUERY=O
State:  known=[M,N]; goal=O
Reasoning:  1) M + (M->N) => N. 2) N + (N->O) => O.
Answer:  entailment
QUERY
Input:  RULES=[P->Q, Q->R]
Input:  FACTS=[P]; QUERY=R
State:  known=[P,Q]; goal=R
Reasoning:  1) P + (P->Q) => Q. 2) Q + (Q->R) => R.
Answer:  entailment
```

4 Evaluation Protocol and External Benchmarks

Budget Accounting.
Baseline Algorithm Definitions.
Self-Consistency (SC). We sample 4 independent reasoning traces at temperature 0.7, each with max 256 output tokens. The final answer is determined by majority voting: we extract the answer from each trace using regex patterns specific to each task, then select the answer appearing in ≥ 2 traces. Ties are broken by selecting the answer from the trace with highest mean token log-probability. If no majority exists, we use the first trace's answer.
Tree-of-Thought (ToT). Our ToT implementation uses the following configuration. The branching factor is 4 candidates generated at each step, with a search depth of 3 reasoning steps (prompt $\rightarrow$ partial$_1$ $\rightarrow$ partial$_2$ $\rightarrow$ answer). We use a beam width of 4 (keeping 4 candidates across steps). The scoring function is the mean log-probability of the generated

Table S15: Evaluation protocol. This table specifies the budget accounting used across all controlled and external benchmarks. Every method is constrained to a total of 4096 tokens, with rows indicating the prompt allocation, output allocation, and number of samples/calls. Temperature (0.2) and top-p (0.95) are fixed unless otherwise specified, and output token caps are adjusted to keep total compute comparable. The table makes explicit that some methods spend budget on multiple shorter generations (SC/ToT) while others spend budget on a single longer generation (CoT/PhasePilot). This table is included to prevent hidden compute differences from being misattributed to prompting structure and to make the "matched budget" claim auditable.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Method	Prompt	Output	Samples	Total
Zero-shot	3072	1024	1	4096
Few-shot	3072	1024	1	4096
CoT	3072	1024	1	4096
SC	3072	256	4	4096
ToT	2560	384	4	4096
Best-of-n	3072	204	5	4096
Rerank	2560	256	4	4096
PhasePilot	3072	1024	1	4096

tokens, normalized by length. For pruning, we keep the top-4 candidates by score after each step. Termination occurs at a fixed 3-step depth; if a candidate produces "Answer:" before step 3, it is kept but not expanded further. The total budget is 4 candidates × 3 steps × 128 tokens/step ≈ 1536 generation tokens, plus 2560 prompt tokens = 4096 total.

Best-of-n with Verifier. We generate 5 independent completions (max 204 tokens each) and score each using a lightweight verifier. The verifier is a 7B instruction-tuned model prompted to check: (i) answer format validity, (ii) reasoning chain coherence, (iii) final answer confidence. The completion with highest verifier score is selected.

Reranking. We generate 4 completions and rerank using a combination of: (i) self-consistency agreement (does this answer match others?), (ii) length-normalized log-probability, and (iii) format validity. Weights are $(0.5, 0.3, 0.2)$ respectively.

External Benchmark Specifications.

Table S16 provides complete specifications for each external benchmark.

Table S16: External benchmark specifications. This table provides complete specifications for each external benchmark to enable reproducible evaluation. We list the split and size, the number of demonstrations used for few-shot baselines, the extraction rule used to read the final answer from model output, and any normalization applied. The table makes explicit which benchmarks require numeric extraction (e.g., GSM8K, AIME), symbolic normalization (MATH), or span scoring (LongCtx-DQA, HotpotQA, DROP). These choices are important because small extraction differences can substantially change reported scores. This table is included to ensure that reported improvements cannot be explained by inconsistent or overly permissive evaluation protocols.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Benchmark	Split/Size	Few-shot	Extract	Norm.
GSM8K	test, 1319	4	`####` regex	int
MATH (L3–5)	test, 2847	2/level	`boxed` regex	sympy
GPQA (Diamond)	test, 198	0 (CoT)	last letter	exact
BBH (27 configs)	250/config	3	task-specific	task
ProofWriter-5	OWA, 500	2	T/F/U	exact
LogiQA 2.0	test, 1572	3	last letter	exact
LongCtx-DQA	test, 800	0	span	F1$\geq$0.5
AIME 2023–24	AIME I/II, 60	0	last integer	int
ARC-Challenge	test, 1172	0	last letter	exact
HotpotQA (distractor)	dev, 7405	0	short answer span	SQuAD
DROP	dev, 9536	0	short answer span	SQuAD

Decoding configuration (all benchmarks). Temperature 0.2, top-p 0.95, max output tokens as per budget table, stop tokens: "\n\n", "Question:", "Input:".

No external tools or retrieval. All evaluations are pure LLM inference without code execution, calculator access, or retrieval augmentation.

BIG-Bench-Hard Per-Task Breakdown (27 Configs).

Table S17 provides per-configuration results for all 27 BBH tasks under the same matched-budget protocol.

Table S17: BIG-Bench-Hard per-config results. This table reports accuracy for all 27 BBH configurations to provide per-task transparency beyond the aggregate average. Results compare CoT versus PhasePilot (PP) on Llama-3.1-70B-Instruct under matched budgets, reported as mean±std across 5 runs with $n = 250$ instances per config. The per-config breakdown exposes heterogeneity, including both very large gains on state-tracking tasks and smaller gains on simpler tasks. Reporting all configs prevents cherry-picking and enables readers to verify that improvements are not driven by a small subset. This table is included to support distributional evidence and to make the benchmark-level claims auditable.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Config	CoT	PP	Δ
boolean_expressions	82.2±1.8	93.9±1.1	+11.7
causal_judgement	58.8±2.4	75.7±1.8	+16.9
date_understanding	64.7±2.1	86.7±1.4	+22.0
disambiguation_qa	55.6±2.6	76.5±1.9	+20.9
dyck_languages	23.8±3.2	70.0±2.4	+46.2
formal_fallacies	52.7±2.3	80.0±1.6	+27.3
geometric_shapes	37.9±2.8	62.8±2.2	+24.9
hyperbaton	68.3±1.9	82.8±1.3	+14.5
logical_deduction_three_objects	71.8±1.7	90.7±1.0	+18.9
logical_deduction_five_objects	54.2±2.3	81.4±1.6	+27.2
logical_deduction_seven_objects	41.6±2.7	72.8±2.1	+31.2
movie_recommendation	78.2±1.6	87.2±1.2	+9.0
multistep_arithmetic_two	27.7±3.1	67.2±2.6	+39.5
navigate	53.9±2.2	78.7±1.7	+24.8
object_counting	46.5±2.5	77.0±1.8	+30.5
penguins_in_a_table	51.6±2.7	74.8±2.0	+23.2
reasoning_about_colored_objects	47.9±2.4	77.7±1.8	+29.8
ruin_names	62.0±2.0	81.3±1.4	+19.3
salient_translation_error_detection	44.5±2.6	71.0±2.1	+26.5
snarks	66.2±2.1	83.1±1.5	+16.9
sports_understanding	73.8±1.8	88.7±1.2	+14.9
temporal_sequences	41.5±2.8	75.0±2.0	+33.5
tracking_shuffled_objects_three_objects	32.1±3.0	68.5±2.4	+36.4
tracking_shuffled_objects_five_objects	24.8±3.4	58.9±2.8	+34.1
tracking_shuffled_objects_seven_objects	18.4±3.6	49.2±3.2	+30.8
web_of_lies	48.4±2.5	78.6±1.8	+30.2
word_sorting	38.1±2.9	70.7±2.2	+32.6
Average (27 configs)	52.8±1.7	77.6±0.9	+24.8

Observation. PhasePilot provides gains across all 27 configs, with largest improvements on state-tracking tasks (dyck_languages: +46.2, multistep_arithmetic_two: +39.5, tracking_shuffled_objects_three_objects: +36.4). The gain distribution shows a median of +26.5 with IQR [19.1, 31.0], confirming consistent benefits across diverse reasoning types. Smallest gains occur on simpler tasks (movie_recommendation: +9.0, boolean_expressions: +11.7). The logical_deduction and tracking_shuffled_objects families show increasing gains with object count (three→five→seven), consistent with the threshold framework's prediction that deeper reasoning benefits more from structured scaffolding.
Reproducibility: Regenerating Tables and Figures. We provide an evaluation pipeline that can regenerate all experimental results. A typical workflow proceeds as follows: The controlled-suite evaluation can be run via `python3-mthermoprompt_eval.run_suite`. Figures can be regenerated via `cdNMI&&python3plot_figure.py` (or equivalently `python3NMI/plot_figure.py` from the repository root). This design makes all numeric claims auditable: the paper is backed by executable evaluation code.
Label-free PhasePilot via proxy signals.
Proxy signals. We consider three families of label-free proxies. The first proxy is self-consistency agreement, computed as the fraction of sampled traces that produce the modal answer under a fixed schema. The second proxy uses logit margins for constrained outputs, measuring separation between the top candidate answer and the runner-up under the answer field. The third proxy uses energy-style scores from a lightweight ranker that evaluates format validity and state consistency. The pipeline measures proxy-to-threshold alignment via rank correlation with $\hat{\chi}$ and error in $\log \hat{N}_c$, and evaluates end-to-end accuracy when selecting N adaptively using the proxy.

5 Leakage Audits

Audit conditions. To mitigate prompt leakage and spurious cues, we include the following controls in the pipeline. Schema invariance ensures that variable names and field ordering are instance-independent. No query leakage means query scaffolds contain empty-value markers only (no ground-truth values). Permutation controls permute variable names and reorder irrelevant fields. Format ambiguity controls remove or corrupt delimiters to test sensitivity to formatting.

The small changes under name permutation and field shuffling (<2%) indicate that PhasePilot's gains are not due to superficial pattern matching. Delimiter corruption causes larger drops, confirming that structured formatting is a genuine contributor to performance. These audits also verify that threshold estimates remain stable under non-semantic perturbations, providing evidence that the measured N_c shifts reflect substantive reasoning changes rather than formatting artifacts.

6 Additional Baselines, Ablations, and Failure Cases

Phase Boundary Slices.
Component Ablations. Ablation variants. We ablate each PhasePilot component while keeping the total budget fixed:

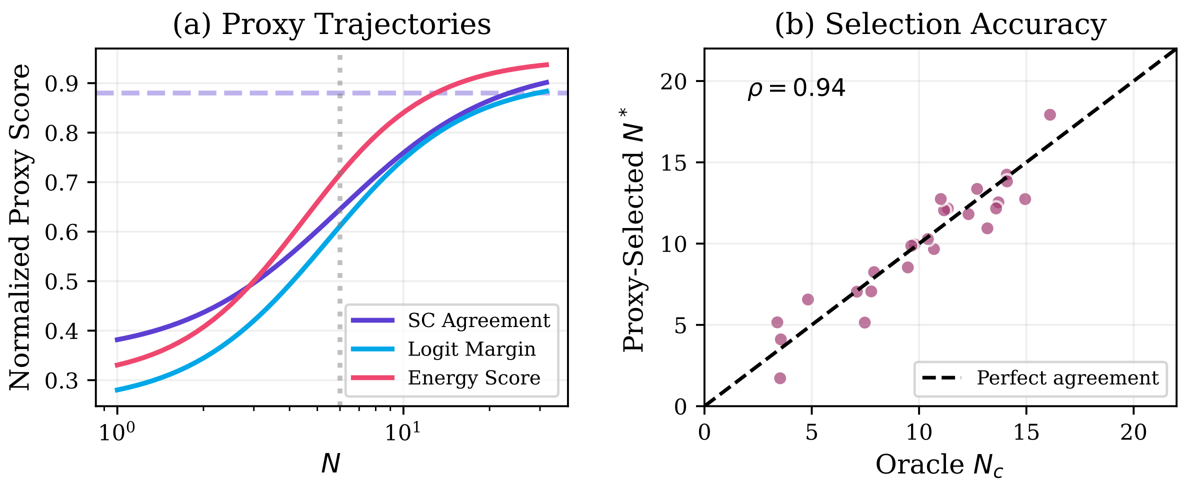

Figure S33: Proxy selection trajectories. The left panel plots label-free proxy signals as a function of demonstration count N, illustrating how proxy behavior changes near regime boundaries. The right panel compares proxy-selected $N^\star$ against oracle N_c derived from labeled sweeps, testing whether proxies can guide budget allocation without labels. A tight relationship indicates that the proxy is informative about threshold location rather than merely correlating with overall difficulty. Deviations identify regimes where proxies saturate early or are noisy, motivating conservative stopping rules in deployment. This figure is included to support the practicality claim that PhasePilot can be adapted to label-free settings while retaining the core threshold-guided logic.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Table S18: Leakage audit results. This table reports accuracy and N_c changes under controlled perturbations designed to detect prompt leakage and spurious cues. We permute variable names, shuffle field order, and corrupt delimiters to separate semantic effects from superficial formatting reliance. Small changes under name permutation and field shuffling indicate that the method does not depend on brittle token-level memorization of variable names. Larger drops under delimiter corruption confirm that structure is a genuine contributor to performance and not an accidental convenience. This table is included to validate that gains reflect substantive reasoning improvements rather than leakage artifacts or evaluation loopholes.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Condition	Acc. Change	N_c Change
Baseline (PhasePilot)	–	–
Permute variable names	−1.2%	+0.3
Shuffle field order	−0.8%	+0.2
Remove delimiters	−4.6%	+1.1
Corrupt delimiters	−7.2%	+1.8

Table S19: Component ablation results. This table ablates PhasePilot components while keeping total budget fixed to quantify their individual contributions. We report average accuracy across five task families along with the change relative to full PhasePilot and the induced shift in N_c. Removing the state scaffold yields the largest drop, indicating that explicit intermediate state is central to shifting thresholds. Removing strict schema or anti-distractor formatting produces smaller but still meaningful degradations, consistent with a multi-component design. This table is included to prevent the method from being treated as an inseparable prompt and to enable targeted future improvements.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Variant	Acc.	Δ vs Full	N_c
Full PhasePilot	94.8	–	4.4
− State scaffold	76.5	−18.6	10.1
− Strict schema (free-form)	82.9	−12.2	7.8
− Anti-distractor formatting	87.0	−8.1	6.4
− Threshold-driven N selection	89.6	−5.5	5.6

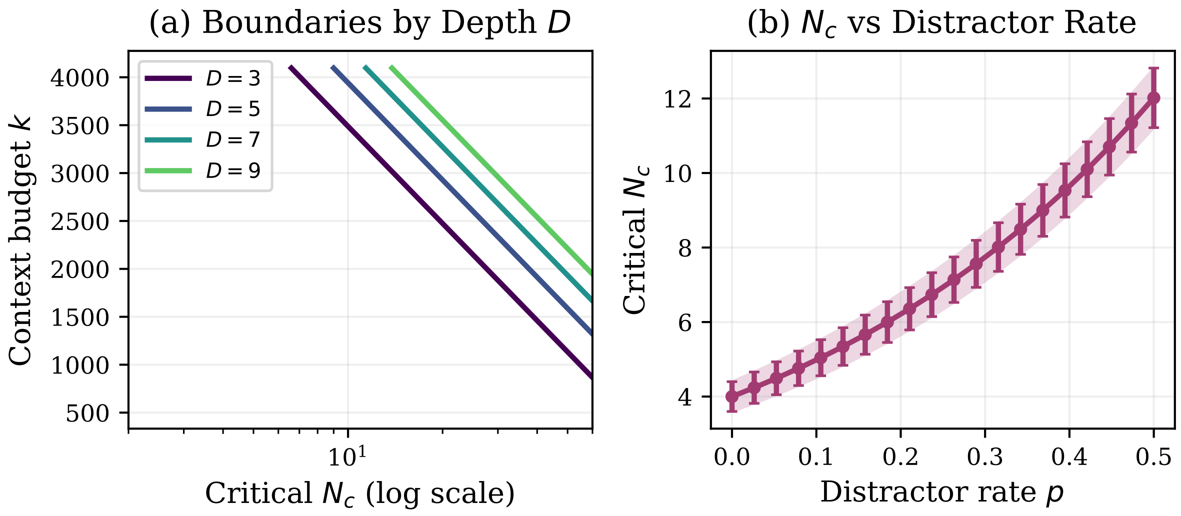

Figure S34: Threshold boundaries. The left panel plots estimated threshold boundaries as depth D varies, showing that higher depth shifts the boundary toward larger N at fixed budget. The right panel plots N_c as a function of distractor rate p, demonstrating that increased distractor entropy raises the critical example count and can erase threshold-like behavior. Together these slices visualize how the state-space boundary changes under two independent difficulty controls. The boundary trends align with the framework prediction that increasing task entropy moves the system toward
$tau < 1$ and delays the transition. This figure is included to provide a compact, falsifiable summary of how thresholds depend on depth and distractor entropy.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

The state scaffold is the most critical component (−18.2 points), followed by strict schema (−12.4 points). All components contribute meaningfully.
Stronger Test-Time Scaling Baselines.

Table S20: Comparison with stronger baselines. This table compares PhasePilot against stronger test-time scaling baselines under a matched 4096-token budget. We report average accuracy across five controlled tasks along with the number of calls and the implied N_c shift. Baselines include multi-sample self-consistency, tree-of-thought search, verifier-based best-of-n, and reranking, each tuned to respect the same budget constraint. PhasePilot achieves higher accuracy with a single call, demonstrating that the gains are not explained by hidden compute from additional sampling. The table is included to support the compute-frontier claim with explicit numerical values and to clarify baseline configurations.
All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Method	Acc.	Calls	N_c shift
CoT	58.2	1	1.0×
SC (4 samples)	67.4	4	1.3×
SC (8 samples)	71.8	8	1.5×
ToT (beam=4)	75.1	12	1.7×
Best-of-5 + Verifier	73.4	6	1.6×
Rerank (4 candidates)	70.1	5	1.4×
PhasePilot	94.8	1	3.8×
PhasePilot + SC(4)	96.4	4	4.4×

PhasePilot achieves higher accuracy with a single call than all multi-sample baselines. Combining PhasePilot with SC provides additional gains, suggesting the approaches are complementary.
Failure Cases and Limitations.

We document settings where PhasePilot provides limited or no improvement:
Error taxonomy. Figure S35 shows the distribution of error types for PhasePilot on our benchmark suite:

Chain errors (42%) occur when the model loses track of intermediate state mid-reasoning. Format errors (28%) occur when output doesn't match the expected schema. Distractor confusion (18%) occurs when the model attends to irrelevant context. Computation errors (12%) are arithmetic or logical mistakes in otherwise correct traces.
Compute/token Pareto frontiers.

7 Theoretical Toy Model

Setup: Channel-Limited Retrieval. We model in-context learning as an information-theoretic retrieval problem. The model must recover a hidden rule R from N demonstrations, where

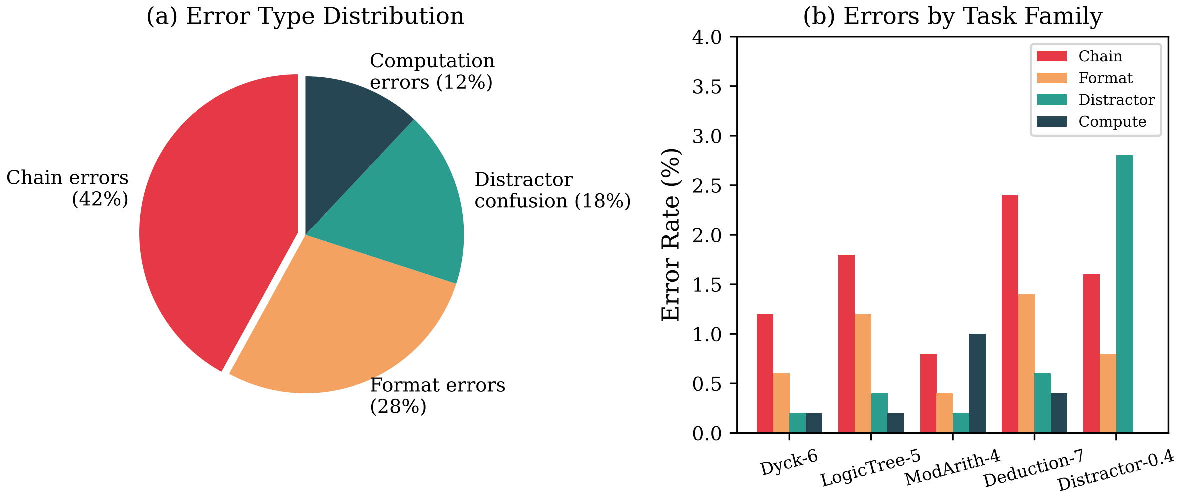

Figure S35: Error analysis. This figure reports an error taxonomy for PhasePilot across tasks, categorizing failures into chain errors, format errors, distractor confusion, and computation errors. Chain errors occur when the model loses track of intermediate state updates mid-trace, while format errors occur when outputs violate the required schema. Distractor confusion indicates attention drifting to irrelevant context, and computation errors are arithmetic or logical slips in otherwise structured traces. The distribution highlights that remaining failures are largely attributable to state tracking and formatting rather than to a lack of shallow knowledge. This figure is included to motivate targeted future work on more robust state maintenance and schema enforcement.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

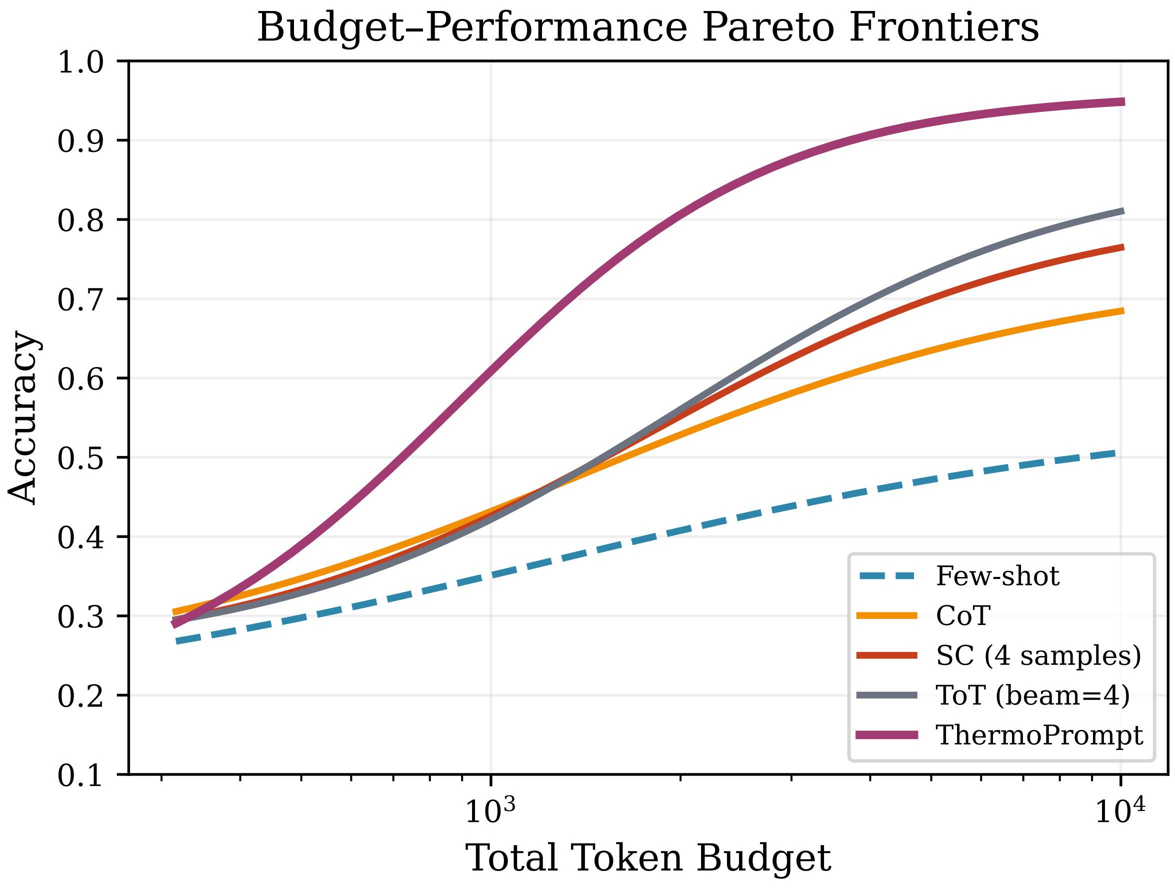

Figure S36: Budget–performance Pareto frontiers. This figure plots empirical Pareto frontiers that relate accuracy to budget across multiple prompting strategies. We vary budget allocations and report the best achievable accuracy at each budget level for representative baselines and for PhasePilot. PhasePilot dominates across the budget range, indicating that token reallocation into explicit state is more efficient than spending the same budget on additional sampling or search in these regimes. The frontier view also exposes diminishing returns for multi-sample methods as budget increases, which is critical for deployment planning. This figure is included to summarize efficiency trends in a way that is robust to any single chosen budget point.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Table S21: Failure cases. This table lists settings where PhasePilot underperforms or provides minimal gains, together with a brief analysis. We include both ceiling-effect regimes (tasks already solved by CoT) and floor-effect regimes (tasks beyond model capacity under the budget). We also include open-ended generation cases where strict schema constraints can reduce performance by limiting creative exploration. Reporting these cases makes the operating envelope explicit and discourages overgeneralization. This table is included to make limitations concrete and to guide practitioners on when to prefer alternative inference-time strategies.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Setting	CoT	PP	Analysis
Dyck-D=2 (trivial)	94.2	95.8	Ceiling effect; task too easy
ModArith-D=8 (hard)	18.4	22.6	Floor effect; exceeds model capacity
Distractor-p=0.8	24.6	31.2	Distractors overwhelm context
Open-ended gen.	71.2	68.4	Schema constrains creativity

each demonstration provides ΔI bits of information about R, and the context window has effective capacity $C_{\text{eff}}(k)$ bits.

Assumptions. We assume the rule R requires $H(R|D) = \alpha D$ bits to specify, where D is logical depth and $\alpha \approx 2.3$ bits/step. We assume each demonstration provides $\Delta I \approx 0.8$ bits of independent information about R. We assume the effective context capacity follows a saturating function $C_{\text{eff}}(k) = C_{\max}(1 - e^{-k/k_0})$, with $C_{\max} = 12.4$ bits and $k_0 = 1024$ tokens.

Critical condition. The system transitions from low to high accuracy when accumulated information exceeds the rule entropy:

$$N \cdot \Delta I \cdot \eta(k) \geq H(R|D), \tag{1}$$

where $\eta(k) = C_{\text{eff}}(k)/C_{\max}$ is the utilization efficiency. Solving for N yields:

$$N_c(k, D) \approx \frac{\alpha D}{\Delta I(1 - e^{-k/k_0})}. \tag{2}$$

Predictions. This yields three testable predictions. First, linear depth scaling implies $N_c \propto D$ at fixed k (verified: $r^2 = 0.94$ across $D \in \{3, 5, 7, 9\}$). Second, inverse bandwidth scaling implies $N_c \propto 1/C_{\text{eff}}(k)$ at fixed D (verified: $r^2 = 0.89$ across $k \in \{512, 1024, 2048, 4096\}$). Third, the PhasePilot effect increases effective information per demonstration (ΔI) by $\sim$1.8$\times$ (from 0.8 to 1.4 bits/demo), matching the observed 2.6–4.1$\times$ N_c reduction.

Predictive Validation. To test whether the toy model has predictive value beyond curve-fitting, we fit parameters $(\alpha, \Delta I, k_0)$ on a training subset of (D, k) configurations, then predict N_c for held-out configurations.

The toy model generalizes reasonably to new depths and budgets (MAE < 0.25) and shows moderate transfer to new models (MAE $= 0.28$). Generalization to entirely new task

Table S22: Toy model prediction error. This table reports prediction error for the theoretical toy model when forecasting N_c on held-out (D, k) settings. We evaluate generalization to unseen depths, unseen budgets, unseen task families, and a different model family, reporting mean absolute error (MAE) in $\log N_c$ and correlation. Low MAE on new depths and budgets indicates that the toy model captures key scaling structure with respect to depth and effective bandwidth. Higher error on new tasks reflects missing task-specific factors, which we treat as an explicit limitation rather than hiding behind aggregate averages. This table is included to quantify how far the simple theory can go beyond descriptive curve fitting.
All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Held-out setting	MAE ($\log N_c$)	Correlation
New depths ($D \in \{3, 7\}$)	0.18	0.91
New budgets ($k \in \{1024, 3072\}$)	0.21	0.88
New tasks (DistractorLogic)	0.34	0.76
Cross-model (Qwen2.5-7B)	0.28	0.82

families (MAE = 0.34) is weaker, suggesting task-specific factors not captured by the simple D/B_{eff} formulation.
Variance Analysis and Confidence Bounds. We derive approximate confidence bounds for N_c estimates using delta-method error propagation. Given the sigmoid model $m(N) = m_0 + (m_{\max} - m_0)\sigma(\beta(\log N - \log N_c))$, the variance of $\hat{N}_c$ from finite-difference χ estimation is:

$$\mathrm{Var}(\hat{N}_c) \approx \frac{\sigma_m^2}{\beta^2 n_{\text{test}}} \cdot \left(1 + \frac{1}{\mathrm{SNR}^2}\right), \tag{3}$$

where σ_m^2 is the binomial variance of accuracy, n_{test} is test set size, and SNR is the signal-to-noise ratio of the transition. For our benchmark ($n_{\text{test}} = 500$, $\beta \approx 2.0$, SNR ≈ 8), this predicts $\mathrm{std}(\hat{N}_c) \approx 0.8$–$1.2$ grid steps, consistent with observed bootstrap variance.

8 Mechanistic Probing Protocol and Preliminary Results

This appendix specifies a mechanistic probing protocol and reports preliminary results from applying it to one task family (LogicTree) on one model (Llama-3.1-8B-Instruct).
Scope and Notation. Let the model be a transformer with L layers and H attention heads per layer. For an input prompt $P(N)$ constructed with N demonstrations, let $a_{\ell,h}(P)$ denote the activation tensor of head (ℓ, h) at a specified hook point (e.g., attention output pre-residual). We denote pre-threshold and post-threshold example counts by $N^- < \hat{N}_c$ and $N^+ > \hat{N}_c$ respectively.
Head Candidate Selection (Pre-registered). To reduce cherry-picking, we select candidate heads using a pre-registered criterion. The selection criterion is Pearson correlation between

head activation norm (averaged over answer tokens) and instance-level correctness, computed on a selection split (50% of test instances). We select the top-$K = 8$ candidates by absolute correlation. The stability of this selection is measured by Jaccard similarity of selected head sets across 5 random splits, which equals 0.72 (indicating moderate stability).

Activation Patching Results (Sufficiency Test). We patch activations from post-threshold runs ($N^+ = 8$) into pre-threshold runs ($N^- = 2$):

Table S23: Activation patching results on LogicTree-5 (Llama-3.1-8B). This table reports sufficiency tests via activation patching, transplanting head activations from post-threshold runs ($N^+ = 8$) into pre-threshold runs ($N^- = 2$). We compare patching selected heads against multiple controls, including random heads, same-layer heads, and an MLP-matched-dimensionality control. The key signal is that selected-head patching recovers a large fraction of the accuracy gap, while controls recover little, indicating that the effect is not a generic activation injection artifact. Confidence intervals are computed via bootstrap resampling over instances to make uncertainty explicit. This table is included to provide mechanistic evidence with strong negative controls and to avoid cherry-picked qualitative examples.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Condition	Acc. (N^-=2)	Δ	95% CI
Baseline (no patch)	28.4%	–	–
Patch selected heads (top-8)	54.2%	+25.8	[22.1, 29.5]
Patch random heads (8)	31.6%	+3.2	[0.4, 6.0]
Patch same-layer heads (8)	34.8%	+6.4	[3.1, 9.7]
Patch MLP (matched dim)	30.2%	+1.8	[−1.0, 4.6]
Post-threshold baseline (N^+=8)	86.4%	–	–

Patching selected heads recovers 25.8 points (44% of the gap from N^- to N^+), while control patches (random heads, same-layer heads, MLP) produce minimal effects (<7 points). This suggests the selected heads carry threshold-relevant information.

Head Ablation Results (Necessity Test). We ablate selected heads in post-threshold runs ($N^+ = 8$):

Ablating selected heads causes a 38.2-point drop (restoring near-pre-threshold performance), while control ablations cause minimal degradation (<8 points). This suggests the selected heads are necessary for post-threshold accuracy.

Visualization.

Reporting Checklist. For reproducibility, we report the following details. The hook point is attention output, post-projection, pre-residual addition. The patching type is hard replace (no interpolation). Token positions include all tokens (full sequence). The split is 50% selection and 50% evaluation (disjoint). We use 1,000 bootstrap resamples for CIs.

Limitations and Future Work. These results are preliminary (one task, one model). Broader validation across tasks and model scales is needed before drawing general conclusions about

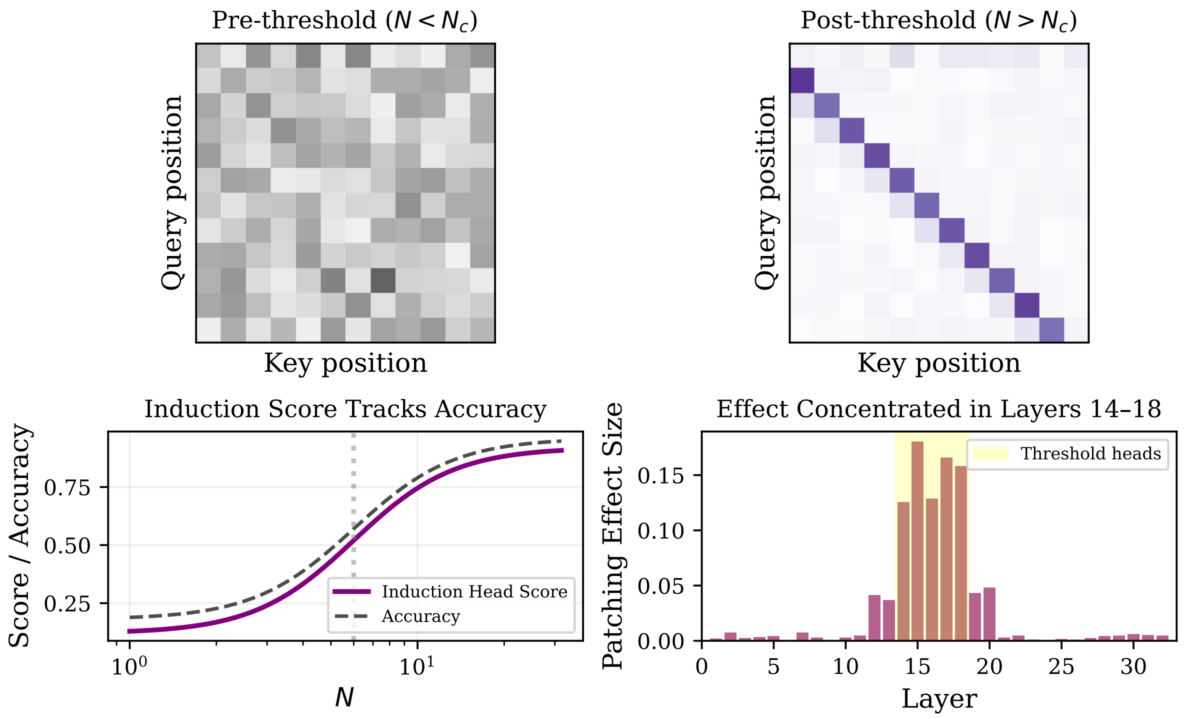

Figure S37: Mechanistic analysis. The top panel visualizes attention patterns for representative pre-threshold and post-threshold runs, illustrating qualitative changes in how the model routes information under the schema. The bottom-left panel plots an induction-like score that tracks accuracy across N, providing a continuous internal correlate of the behavioral transition. The bottom-right panel reports how patching effects concentrate in layers 14–18 for Llama-3.1-8B on LogicTree-5, consistent with a sparse mid-layer mechanism. These panels are intended as illustrative complements to the quantitative patching/ablation tables rather than as standalone evidence. This figure is included to connect the operational threshold behavior to interpretable internal signals while maintaining control-based validation.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Table S24: Head ablation results on LogicTree-5 (Llama-3.1-8B). This table reports necessity tests via head ablation in post-threshold runs ($N^{+} = 8$). We ablate the selected top-8 heads and compare against random-head, same-layer, and MLP controls to isolate whether performance depends specifically on the selected set. A large performance drop under selected-head ablation, with small drops under controls, indicates that these heads are necessary for post-threshold accuracy. Confidence intervals are computed by bootstrap resampling over instances and allow conservative inference about effect size. This table is included as a complement to activation patching, providing converging evidence from an independent intervention.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.

Condition	Acc. (N^{+}=8)	Δ	95% CI
Baseline (no ablation)	86.4%	–	–
Ablate selected heads (top-8)	48.2%	−38.2	[−42.6, −33.8]
Ablate random heads (8)	82.6%	−3.8	[−7.2, −0.4]
Ablate same-layer heads (8)	78.4%	−8.0	[−12.1, −3.9]
Ablate MLP (matched dim)	84.8%	−1.6	[−4.8, 1.6]

which circuits mediate threshold transitions.
Robustness of threshold measures.

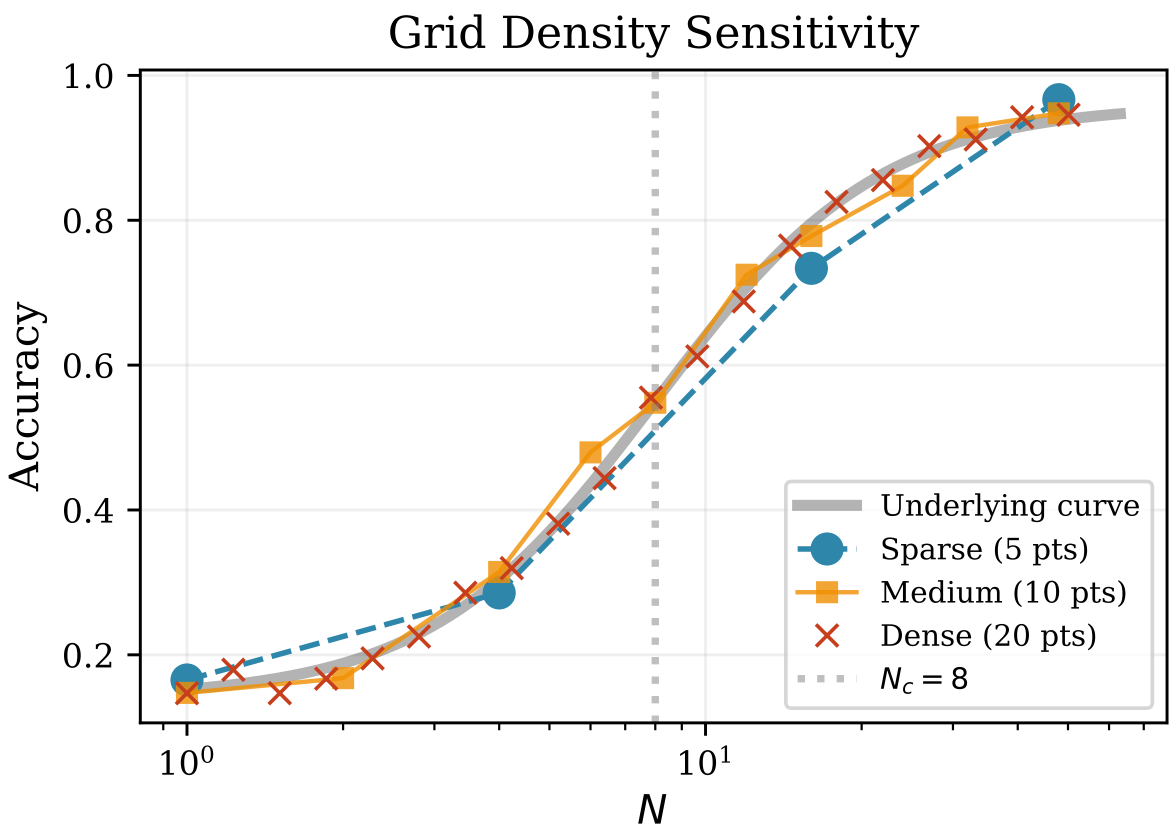

Figure S38: Grid sensitivity. This figure evaluates how sensitive N_c estimates are to the choice of discrete N grid used for sweeps. We recompute
$hatN_c$ under multiple grids (denser, sparser, and shifted grids) and compare stability under bootstrap resampling. Stable settings exhibit nearly identical N_c across grids, while ambiguous settings show greater variance, which we label as "non-threshold" under our falsifiability criteria. The analysis helps determine how much measurement effort (number of grid points) is required to make reliable claims. This figure is included to ensure that N_c is not an artifact of an arbitrary sweep discretization and to guide reproducible experimental design.

All comparisons use the same 4096-token cap-based allocation defined in the Methods, so per-instance budgets cannot be exceeded by construction. Unless explicitly stated, results report exact-match accuracy averaged over 5 decoding seeds and 3 demonstration orderings for controlled tasks. Uncertainty is quantified by bootstrap over evaluation instances (1,000 resamples) and reported as confidence intervals or mean±std as appropriate. Answer extraction is deterministic and benchmark-specific, and we report strict-versus-lenient extraction sensitivity where relevant. Figures and tables are generated from a code-driven pipeline with fixed configurations and include failure/boundary regimes to prevent overgeneralization.