Contrastive Thinking Decoding: Steering Answer Generation in Reasoning Models

Taehyeon Kim[◇*] **Hyunsoo Lee**[♠] **Yu Jin Kim**[♠] **Youngsoo Jang**[‡,†] **Moontae Lee**[♠,§,†]

[◇]Holiday Robotics [‡]UNIST [♠]LG AI Research [§]University of Illinois Chicago

[†]Co-corresponding authors.

Abstract

Large reasoning models separate inference into an explicit *thinking* phase followed by a final *answer* phase, yet how the answer phase uses the reasoning trace remains unclear. We present a systematic study of this trace-to-answer transition across multiple LRMs and find that (i) final answers diverge from correct reasoning traces at rates between 6% and 48%, even when traces terminate cleanly at full budget, (ii) naively copying the trace answer underperforms standard decoding in some settings, revealing that the answer phase has independent corrective capacity, and (iii) the same phenomenon extends beyond structured tasks to settings where reasoning traces and final answers can diverge under social-pressure framing (sycophancy). These findings motivate Contrastive Thinking Decoding (**CTD**), a training-free, single-model decoding method that contrasts answer-phase logits under the primary reasoning trace against those from a deliberately degraded noisy trace. **CTD** selectively amplifies trace-consistent tokens while preserving the answer phase's corrective capacity, concentrating CTD-internal joint mass on the both-correct cell (T✓A✓) without inducing trace-correct-to-answer-wrong drift relative to the base distribution. Across math reasoning, science, code, social-pressure resistance (sycophancy), and a broad knowledge benchmark (MMLU), **CTD** reduces trace-answer disagreement and improves the accuracy–compute trade-off without parameter updates or auxiliary models at decoding time.

1 Introduction

Large reasoning models (LRMs) have recently advanced complex reasoning tasks such as arithmetic, commonsense reasoning, code [Jain et al., 2024], and symbolic tasks [Rae et al., 2021], often by a wide margin [Guo et al., 2025, Yu et al., 2025]. At inference time LRMs externalize an explicit *thinking* phase before producing the answer (Jaech et al. [2024], OpenAI [2025], Guo et al. [2025], *inter alia*). A key difference from earlier approaches to complex reasoning is the explicit optimization of this thinking phase. Prior work has largely relied on human supervised learning to elicit reasoning capabilities in large language models (LLMs), including prompting methods such as few-shot prompting [Brown et al., 2020, Wei et al., 2022, Zhou et al., 2022, Chen et al., 2022], carefully designed system and user instructions [Kojima et al., 2022, Sahoo et al., 2024], and training models with substantial chain-of-thought (CoT) reasoning data [Chung et al., 2024, Muennighoff et al., 2025]. In contrast, LRMs employ reinforcement learning with verifiable rewards as black box optimization over thinking traces to reach correct answers during training. Recently, optimization of such traces in LRMs has been largely studied to improve reasoning ability [Shrivastava et al., 2025, Zhang et al., 2025b, Zhao et al., 2025, Wang et al., 2025].

[*]Work primarily done while at LG AI Research.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

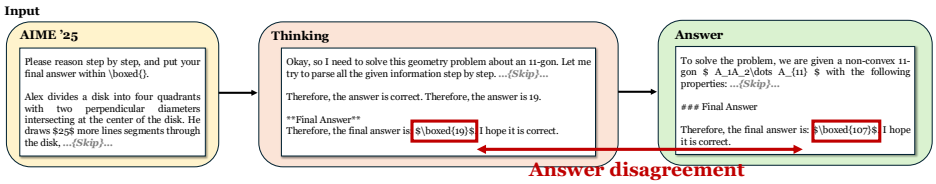


Figure 1: An example of trace-answer disagreement from Qwen3 8B. The thinking trace arrives at a correct answer, yet the answer phase recomputes with extra steps and outputs a different value. A full example is described in Appendix I.

While the thinking phase is important, the transition from a completed reasoning trace to a final answer remains under-explored. One recurring observation is *trace-answer disagreement*: even when the trace is high quality, the answer can inject unsupported steps or flip the final choice. When the trace is partial or inconsistent, the answer often extrapolates beyond the evidence (Figure 1, §2.1). A likely cause is that answer decoding maximizes next token likelihood under the full context, which can place high mass on generic continuations, boilerplate, or spurious tokens that are only weakly tied to the trace.

The answer phase is not merely a passive readout of the trace: it has *independent corrective capacity*. Naively copying the last boxed answer from the trace can underperform standard decoding in some settings, because the answer phase sometimes corrects trace errors (§2.2). This motivates a decoding-time control that steers the answer toward evidence in the trace while preserving the answer phase's corrective ability, rather than rigidly enforcing trace-copying. Our contributions have two components: a systematic study of how answers diverge from traces, and a decoding-time method built on the resulting insights.

- **Trace-answer disagreement.** As thinking budget grows, the trace solves more items, yet the answer still strays from it. Conversely, even a partial or noisy trace can yield a correct answer. This dissociation shows that improving the trace and committing to a trace-consistent answer are distinct problems (§2.1).
- **The oracle-copy gap.** Naively inheriting the trace answer ("oracle copying") underperforms standard decoding in some model–task settings, revealing that the answer phase performs useful error correction. This gap shapes how any steering method should operate: soft anchoring rather than hard copying (§2.2).

Motivated by these findings, we introduce Contrastive Thinking Decoding (**CTD**), a test-time framework that uses the thinking phase as a handle for answer-phase control. **CTD** derives a noisy reference from the thinking path using a thinking prompt and applies token-level logit extrapolation between this reference and the primary trace during the answer phase (Figure 2). Critically, **CTD** steers toward trace consistency without rigidly enforcing it, preserving the answer phase's ability to correct trace errors. This is why **CTD** can outperform standard decoding, naive trace-copying, and budget-only control (§2.2, §F.12).

Unlike prior contrastive decoding [Li et al., 2022, Liu et al., 2021, O'Brien and Lewis, 2023] that pairs a strong model with an auxiliary weaker model, **CTD** performs contrast within a single LRM using the thinking phase itself as the source of contrast. We are not aware of prior work that derives the contrastive negative from a perturbed reasoning trace within the same LRM: instructive decoding [Kim et al., 2023] perturbs the instruction rather than the trace, and prior contrastive decoding [Li et al., 2022, O'Brien and Lewis, 2023] relies on an auxiliary weaker model.

Across experiments, **CTD** reduces trace-answer disagreement to a decoding-time control problem. Additionally, **CTD** shifts the accuracy-token frontier, enabling better trade-offs between test-time compute and correctness. In particular, **CTD** attains accuracy-token trade-offs comparable to several training-based approaches while remaining purely inference-time and lightweight, and operates at single-sample inference cost (complementary to, not a replacement for, self-consistency). At 32B (Qwen3), **CTD** improves GPQA Diamond Pass@1 (McNemar $p<0.001$) and narrows the trace-answer gap on sycophancy under social pressure [Sharma et al., 2024]; MMLU is preserved with a majority of subjects winning or tying (§4.4). Qualitative examples can be found in Appendix I.

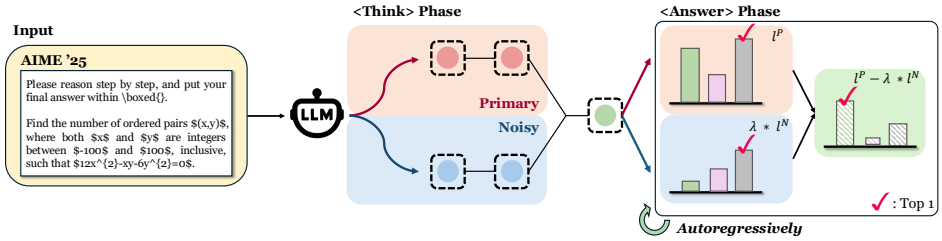


Figure 2: Overview of contrastive thinking decoding (**CTD**).

Table 1: Thinking-answer disagreement under fixed thinking budgets (avg. over 16 samples). As budget increases, solved-in-trace rates rise but non-trivial disagreement remains, indicating that trace quality and answer correctness are separable.

Model	Budget	Solved in trace (%)	Disagreement (%)	Pass@1 (%)	Avg think	Avg answer	Avg total
R1-Distill Qwen 7B	4K	19.69	83.33	49.58	3100	4698	7798
(AIME'24)	8K	34.17	56.67	49.48	4784	1682	6466
	16K	45.73	36.67	50.00	8927	723	9650
	32K	46.14	28.17	51.46	9655	844	10499
Qwen3 1.7B	4K	6.42	84.22	16.90	2871	2323	5194
(AIME'25)	8K	16.04	67.41	20.53	6022	1057	7080
	16K	23.54	43.33	25.50	9729	910	10639
	32K	33.33	15.00	34.79	14160	916	15076

2 How answers diverge from reasoning traces

2.1 Measuring trace-answer disagreement

We empirically observe that the answer can diverge from the thinking trace. Figure 1 shows a representative pattern: the thinking trace arrives at `\boxed{19}` after a derivation, yet the answer phase recomputes with extra steps and outputs `\boxed{107}`.

Metric definition. We generate a thinking trace once under a given budget, keep it fixed, and resample only the answer phase under different seeds. An item is *solved in trace* when the thinking trace contains a correct `\boxed{}` (we extract the *last* `\boxed{}` before `</think>`, so intermediate revisions are superseded). *Disagreement* holds when the answer `\boxed{}` differs from the trace `\boxed{}`. Instances where the trace produces no `\boxed{}` are excluded from the disagreement denominator. For a fixed final answer, traces with different derivations but the same extracted conclusion receive the same score under this metric.

Addressing truncation and clean-trace artifacts. Budget-forced traces may terminate mid-reasoning, inflating disagreement. At the 32K budget, clean traces (those that terminate naturally and contain a boxed answer) cover over 91% of items across audited models, with one exception: Qwen3-1.7B reaches only 53–62% on AIME because smaller Qwen3 variants inconsistently emit `\boxed{}` inside the trace, and we excise it from clean-trace claims.[2] Truncated traces and traces without a boxed answer are excluded from the disagreement denominator. Restricted to the clean-trace subset, disagreement spans 6.4–48.2%, essentially the full-set band reported in Table 1; the gap is not a truncation or extraction artifact. We report at 32K because trace completion is highest there; lower-budget cells (4K–16K) in Table 1 are retained as descriptive measurements of the same metric, with full per-budget clean-trace rates in Table 21.

Results. Table 1 shows that larger budgets raise solved-in-trace rates, yet non-trivial disagreement persists. Pass@1 increases more slowly than solved-in-trace, indicating that trace quality and answer correctness decouple. Disagreement rates range from 6% (Qwen3-8B, AIME'24, 32K) to 48% (Exaone4-1B, AIME'24, 32K) across model families. Disagreement is a descriptive measurement of how often the answer phase commits to a different conclusion than the trace; it is not by itself a

[2]Per-model clean rates are in Table 21. Among the 31 multi-boxed Qwen3-8B traces at 32K, 30 have the last `\boxed{}` matching the final answer, so intermediate-answer aliasing affects at most one instance.

Table 2: The oracle-copy gap (AIME'24, 32K). For R1-7B, standard decoding outperforms naive trace-copying. **CTD** exceeds both.

Model	Oracle	Base	CTD
Qwen3-8B	68.8	75.9	**77.7**
R1-Distill 7B	48.5	51.5	**57.7**

Table 3: Four-way decomposition under **CTD** (AIME'24, 16 runs per problem, full budget, **CTD**-Null). Recovery (T×A✓) outnumbers drift (T✓A×) by an order of magnitude in every model with per-instance four-way logging. Counts are CTD-internal and not a Base-vs-CTD significance test.

Model	Drift (T✓ →A×)	Recovery (T× →A✓)
Qwen3-8B	1 (0.4%)	6 (2.7%)
R1-Distill 7B	2 (1.5%)	46 (13.1%)
Exaone4 1.2B	0 (0.0%)	70 (19.7%)
Pooled	25	341

faithfulness claim, and the operational quantity that drives **CTD**'s gains is the per-instance *recovery vs. drift* split (Table 3), not the disagreement rate alone.

Finding 1: Trace-answer disagreement is pervasive

Even as *solved in trace* increases with budget, a non-trivial disagreement rate remains (between 6% and 48% across the LRMs we evaluate). Conversely, the answer can be correct even when the trace is partial. This gap indicates that improving the trace and committing to a trace-consistent answer are distinct problems.

2.2 The oracle-copy gap: answer-phase error correction

A natural question is: how much accuracy would we gain by simply copying the trace answer? We implement a *trace oracle* that takes the last `\boxed{}` in the trace as the final answer.

Table 2 reveals a striking pattern: for R1-7B, the trace oracle (48.5%) underperforms standard decoding (51.5%). The answer phase sometimes corrects errors that the trace committed to. Standard decoding allows uncontrolled deviation, sometimes helpful (correcting trace errors) and sometimes harmful (overriding correct traces).

CTD resolves this by performing *soft anchoring*: it steers toward trace consistency via contrastive logits without rigidly enforcing trace-copying. The four-way decomposition (Table 3) confirms this: under **CTD**, recovery (T×A✓) outnumbers drift (T✓A×) by an order of magnitude in every per-instance-logged condition, with drift bounded at or below 1.5% on those conditions.

Finding 2: The answer phase has independent corrective capacity

Naïve trace-copying can underperform standard decoding because the answer phase sometimes corrects trace errors. **CTD** achieves CTD > Base > Oracle by softly steering the answer toward trace-consistent tokens without enforcing copying; per-cell numbers are in §5.1.

2.3 Budget control alone is insufficient

We control test-time compute by placing an explicit token budget on the `<think>...</think>` region and closing it at the next step boundary. Training-free budget control is itself a strong accuracy–compute baseline, comparable to several budget-aware training-based methods (§F.12). Yet Table 1 shows it does not eliminate drift: small budgets push extra reasoning into the answer ("over-answering"), large budgets concentrate verification inside thinking, and trace–answer disagreement persists across the range. Explicit answer-phase control is therefore needed beyond budget tuning.

Thinking prompts as structured trace perturbations. We define a small space of *thinking prompts* injected only inside `<think>...</think>`: EMPTY (no content), NULL (minimal placeholder `.\n\n`), SP (shortcut reasoning), PP (5-step plan), and OPPOSITE (deliberately wrong reasoning); full templates are in Appendix H. These prompts materially steer answer-phase behavior (NULL and OPPOSITE both produce traces with minimal reasoning content, yielding weak trace accuracies but

strong negative references for contrast), and motivate using prompt-induced traces as structured *noisy references* for **CTD**.

3 Contrastive thinking decoding

Building on the findings in §2, we propose Contrastive Thinking Decoding (**CTD**), a training-free decoding method that steers the answer toward the trace without rigidly enforcing trace-copying, using a single model and no parameter updates.

Design. **CTD** operates with two trace-conditioned contexts from a *single* LRM: a *primary* trace s^P (vendor default reasoning) and a *noisy reference* trace s^N (injected thinking prompt, e.g., NULL or OPPOSITE). During the answer, token-level scores under both traces form a contrast that enhances primary-supported tokens while downweighting noisy-reference tokens (Figure 2).

Rationale. Answer decoding reflects a mixture of (i) trace-conditioned reasoning cues and (ii) generic language priors (boilerplate, common continuations). Noisy references suppress the former while retaining the latter. The extrapolation $\tilde{\ell}_t = \ell_t^P - \lambda \ell_t^N$ acts as a filter that downweights the shared generic prior and preserves trace-specific tokens. Because **CTD** applies contrastive reweighting with $\lambda < 1$ rather than hard trace-copying, it preserves the answer phase's corrective capacity (§2.2), explaining why CTD $>$ Base $>$ Oracle.

Contrastive rule. Let $\ell_t^P, \ell_t^N \in \mathbb{R}^{|\mathcal{V}|}$ be the logits at step t conditioned on s^P and s^N:

$$\ell_t^P = \log p_\theta\big(\cdot \mid x, s^P, y_{<t}\big), \qquad \ell_t^N = \log p_\theta\big(\cdot \mid x, s^N, y_{<t}\big).$$

Contrastive logits: $\tilde{\ell}_t = \ell_t^P - \lambda\, \ell_t^N$, then decode with $q_t = \mathrm{softmax}\big(\tilde{\ell}_t/\tau\big)$, $y_t \sim \mathrm{Dec}(q_t)$. Here $\lambda \geq 0$ controls the contrast strength and $\tau > 0$ sets the sharpness; $\lambda = 0$ recovers vanilla decoding.

Hyperparameters and compute. We set λ=0.15 and τ once, reused across all datasets and models (full hyperparameter values in Appendix H); Pass@1 is flat over $\lambda \in [0.05, 0.30]$ across the LRMs we evaluate (Table 13), so the global default does not require per-model tuning. At each answer-phase step **CTD** evaluates the primary-trace-conditioned and noisy-trace-conditioned forward passes in a single batched call, so the doubled answer-phase compute does not double wall-clock latency. The doubled answer-phase forward is bounded by the answer-phase fraction of total tokens; on Qwen3-8B AIME'24 at 32K this fraction is 5.4% (§4.3), and the answer-phase fraction (a single-digit percentage of total compute at large thinking budgets) sets the upper bound on marginal overhead across the configurations we evaluate. End-to-end latency on vLLM 0.10.1 is 97.1% (Qwen3-8B) and 90.5% (R1-7B) of base, because the two answer-phase passes batch in parallel and the noisy reference is a fixed prefill (Appendix H). A formal variational derivation is in Appendix G.

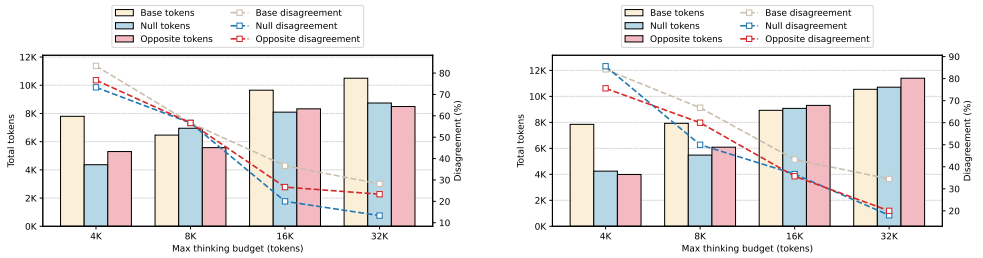
4 Experiments

4.1 Experimental setup

We evaluate four families of LRMs spanning parameter scales and training recipes: Qwen3 1.7B and 8B [Yang et al., 2025], OpenThinker3 7B [Guha et al., 2025], Exaone 4.0 1.2B [Research et al., 2025], and R1-Distill Qwen 1.5B and 7B [Guo et al., 2025]. Benchmarks cover MATH500 [Hendrycks et al., 2021, Lightman et al., 2023], AIME '24 [MAA, 2024], AIME '25 [MAA, 2025], LiveCodeBench[3], and GPQA Diamond [Rein et al., 2024]. We employ vLLM [Kwon et al., 2023] and HuggingFace [Habib et al., 2023] with customizations.

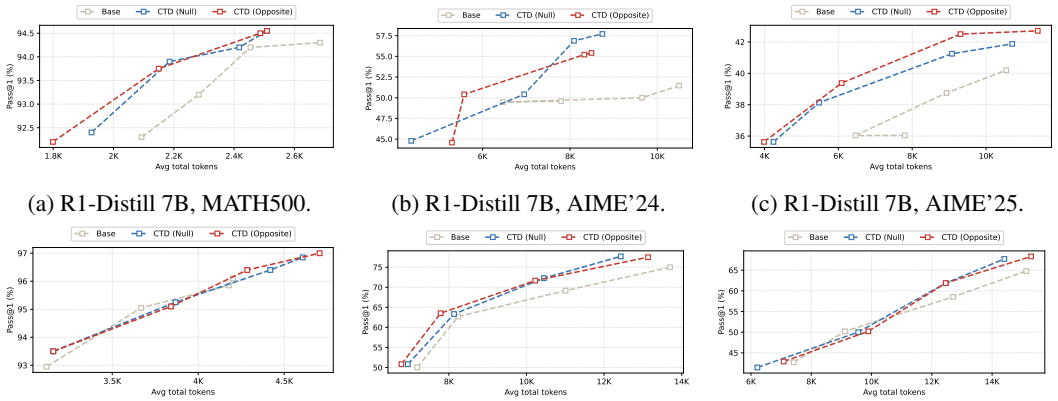
For **CTD** we form one primary trace and one noisy reference trace, then apply token-level extrapolation during the answer. Results on MATH500 are means over 4 runs; AIME (n=30 problems), LiveCodeBench, and GPQA Diamond (n=198) are means over 16 runs. Self-consistency [Wang et al., 2022] (Maj@N at matched total compute) is a complementary axis to **CTD** and is left to future work; the comparisons in this paper are between single-sample decoders. The answer phase uses greedy decoding for both baseline and **CTD**; we extract the final answer as the last `\boxed{}` in the

[3] Slice from March 25 2025 to April 15 2025



(a) AIME'24. (b) AIME'25.

Figure 3: **CTD** vs. Base disagreement rates on AIME'24/'25 (R1-Distill Qwen 7B, NULL/OPPOSITE).



(a) R1-Distill 7B, MATH500. (b) R1-Distill 7B, AIME'24. (c) R1-Distill 7B, AIME'25.

(d) Qwen3 8B, MATH500. (e) Qwen3 8B, AIME'24. (f) Qwen3 8B, AIME'25.

Figure 4: Accuracy vs. total compute (primary thinking + noisy-reference prefill + answer phase $\times$ 2 forward passes). **CTD** shifts the frontier outward on AIME at every measured operating point and preserves it on near-saturated MATH500.

answer phase, and the trace answer (used for the disagreement metric) as the last `\boxed{}` before `</think>`. For the noisy reference we select NULL and OPPOSITE. We use vLLM 0.10.1 [Kwon et al., 2023] on a single H200; the budget cutoff closes the `<think>` block at the next `.\n\n` sentence boundary once the token budget is approached. The contrastive hyperparameters are λ=0.15 and a single τ value reused across all datasets and models (Appendix H).

4.2 Reducing trace-answer disagreement

We measure disagreement as defined in §2.1: given a fixed trace containing a correct `\boxed{}`, we check whether the final answer differs. Figure 3 shows that **CTD** consistently reduces disagreement across budget levels.

4.3 Accuracy–compute Pareto frontiers

Figure 4 shows accuracy versus fully-accounted compute across budgets, where the x-axis sums the primary thinking trace, the noisy-reference prefill, and the answer phase under two forward passes per token. On AIME, **CTD** shifts the frontier outward at every measured operating point; on MATH500 (near saturation), the frontier is preserved.

Compute accounting. End-to-end latency on vLLM 0.10.1 (single H200) is 97.1% of base for Qwen3-8B and 90.5% for R1-Distill 7B (Appendix H): **CTD** is wall-clock-faster than base on both, because the two answer-phase passes batch in parallel and **CTD** emits 240–1,270 fewer total tokens than base at every measured budget. Mechanistically, **CTD** reuses the same thinking trace (identical

thinking tokens); the additional cost is (i) prefilling a fixed noise string ($\sim$2 tokens for NULL, $\sim$100 tokens for OPPOSITE), a single batched operation, not autoregressive; and (ii) each answer token requires two forward passes batched on the same GPU. At budget 32K, the answer-phase fraction of total tokens is 5.4% for Qwen3-8B AIME'24 and 7.5% for R1-Distill 7B AIME'24, so the doubled answer-phase pass adds at most 5–8% FLOPs there; across the configurations we measure, marginal overhead is bounded above by the answer-phase fraction, a single-digit percentage at large thinking budgets. **Matched total compute.** Charging both phases, total **CTD** compute at the 32K-budget setting is $\approx$25K tokens for Qwen3-8B AIME'24, compared to $\approx$32K tokens for base at the same budget; at this matched-compute point of $\sim$25K total tokens, **CTD** reaches Pass@1 77.7 versus base's 75.4 at its own 32K-budget point.

4.4 Generalization across tasks and scales

Unconstrained budgets and scalability. Without explicit budget control, **CTD** improves AIME Pass@1 across the three smaller models we test (Exaone4 1.2B, R1-Distill 1.5B, Qwen3 1.7B; full numbers in Appendix F). At 32B, the scaling evidence rests on GPQA Diamond (Table 4), which has an order of magnitude more items than AIME and supports a statistically significant gain at 32B; on AIME'24 at 32B **CTD** shows a sub-one-problem delta (Appendix F) which we treat as preservation rather than improvement. Within Qwen3, disagreement decreases with scale, and gains shrink correspondingly on saturated benchmarks while remaining substantive on GPQA at 32B.

Table 4: Beyond math and code (avg. over 16 runs). GPQA McNemar $p<0.001$ at both 8B and 32B (n=198).

Task	Model	Base	CTD
GPQA	8B	57.6	**63.6** †
GPQA	32B	68.2	**72.7** †
Sycophancy	8B	30.4	**32.0**
Sycophancy	32B	36.4	**38.8**
MMLU	8B	82.8	**83.1**

†: McNemar $p<0.001$.

Science, social-pressure resistance, and general knowledge. Table 4 summarizes results beyond math and code. On GPQA Diamond [Rein et al., 2024], **CTD** improves accuracy at both 8B and 32B with McNemar $p<0.001$ in both cases. In sycophancy evaluations [Sharma et al., 2024], **CTD** narrows the trace–answer gap under social-pressure framing at both scales (qualitative side-by-side examples in Appendix I); we report the aggregate accuracy delta only, not a per-instance T✓A× rate, since per-trace conclusions were not scored at scale. The sycophancy result shows the trace–answer split is not a math-formatting artifact; we do not interpret it as a safety, alignment, or faithfulness claim (see Appendix A for the scope statement). On MMLU, the majority of subjects win or tie under **CTD** at the recommended setting, indicating no regression on general knowledge (qualitative example in Appendix I). On LiveCodeBench, OPPOSITE improves Pass@1 at most operating points and dominates over the full sweep, while NULL occasionally regresses; we recommend OPPOSITE as the default for code (full results in Appendix F).

4.5 Instance-level gains and decoding baselines

Instance-level win-rate breakdowns (Qwen3-8B AIME, 16 runs per problem) are deferred to Appendix F: **CTD** converts a meaningful subset of previously unsolved instances with larger gains on harder AIME'25, while loss rates remain stable and ties dominate on easier items. We compare against contrastive decoding [Li et al., 2022] (auxiliary weaker model) and instructive decoding [Kim et al., 2023] (instruction control cue) in Table 5: **CTD** outperforms both using a single model, with particularly strong Pass@16 gains over the strongest decoding baseline. Self-consistency can also be applied to **CTD** outputs by voting over independently generated samples. The Maj. column in Table 16 reports the fraction of problems for which the most frequent sampled answer is correct.

Table 5: Decoding baselines (Qwen3-8B AIME'24, max budget).

Method	P@1	P@16
Contr. Dec. [Li et al., 2022]	76.5	86.7
Instr. Dec. [Kim et al., 2023]	75.7	86.7
CTD (NULL)	**77.7**	**90.0**
CTD (OPP)	77.5	**93.3**

4.6 Robustness and ablations

Designed vs. random references. An arbitrary RANDOM prompt still outperforms base on R1-7B AIME'24, confirming the contrastive mechanism is structural. Designed prompts give larger gains, with NULL adding 9.2 points and OPPOSITE adding 6.9 over base on the same configuration, showing that structured perturbation better targets trace-inconsistent tokens.

Vocabulary truncation. Top-k=128 masking during answer-phase extrapolation preserves accuracy (see Appendix F).

Gains correlate with disagreement rate. A per-step contraction bound on the unsupported-token mass under **CTD** is given by Proposition 2 in Appendix G, with Proposition 1 providing the variational characterization that motivates the contrastive form. Empirically, gains are monotone in the disagreement rate across the five LRMs we evaluate (Spearman ρ=1.0 on AIME'24 at 32K), as predicted by the per-step contraction bound; per-model recovery/drift numbers for the three models with per-instance four-way logging are in Table 6.

5 Analysis

This section consolidates the key analytical results that characterize when and why **CTD** works, addressing the central empirical questions about trace-answer disagreement.

5.1 Asymmetric recovery vs. drift under CTD

The four-way decomposition in Table 3 pools across 14 model-benchmark-noise cells covering the three models with per-instance four-way logging (Exaone4 1.2B, R1-Distill 7B, Qwen3-8B), multiple benchmarks, and both NULL/OPPOSITE: 341 recoveries against 25 drifts. Table 6 shows this asymmetry holds at the per-model level: no individual condition shows drift greater than recovery. The percentages normalize drift by correct traces and recovery by incorrect traces. In Table 6, $\Delta\text{Acc} = 100(\text{recovery} - \text{drift})/480$ measures the change from trace accuracy to answer accuracy within **CTD**. We report this ratio as a structural property of the CTD output distribution rather than as a Base-vs-CTD significance test, because (i) Base outputs also exhibit very low drift in the same 4-way decomposition, so the operational difference is not a drift gap but the joint shift toward T✓A✓ (§5.5), and (ii) AIME's small per-benchmark item count makes within-pipeline McNemar tests structurally underpowered for the few-percent effects observed. We therefore use GPQA Diamond, which has an order of magnitude more items, as the load-bearing significance result (§5.4).

Table 6: Per-model recovery-to-drift under **CTD** on AIME'24 (CTD-Null, 480 instances/cell, budget 32K), restricted to the three models with per-instance four-way logging on this condition. Drift ≤1.5%; recovery up to 19.7%. Base also exhibits low drift in the same decomposition (App B); the contrast is the joint shift toward T✓A✓, not a drift gap.

Model	Dis.	Drift	Recov.	ΔAcc
Exaone4 1.2B	48.2	0 (0.0%)	70 (19.7%)	+14.6
R1-Distill 7B	33.3	2 (1.5%)	46 (13.1%)	+9.2
Qwen3 8B	6.4	1 (0.4%)	6 (2.7%)	+1.0
Pooled (14 cond.)	—	25	341	**13.6:1**

Why CTD does not amplify wrong reasoning. **CTD** acts only on the answer phase with the trace fixed. On instances where the trace is wrong (T×), **CTD** either recovers the correct answer or leaves it wrong: it does not convert T×A✓ into T×A×. The only risk is T✓A✓→T✓A× drift, which occurs at ≤1.5% across the three models with per-instance four-way logging (Table 6). A plausible account, consistent with but not directly measured by our experiments, is that the contrastive signal scales with the primary-vs-noisy logit divergence and amplifies trace-*consistent* probability mass: when the trace is correct, this pushes toward the right answer; when the trace is wrong, the divergence tends to be smaller, allowing the answer phase's independent computation to dominate.

5.2 Gains scale predictably with disagreement

A per-step contraction of unsupported-token mass under **CTD** is given by Proposition 2 in Appendix G; the analogous integrated-accuracy bound is not closed-form, but the empirical pattern

follows the prediction. Across the five LRMs we evaluate on AIME'24 at 32K, the gain is monotone in the disagreement rate (Spearman ρ=1.0); we report only the ordering and do not interpret it as a continuous correlation. Three readings follow: (i) gains are modest when disagreement is low, which is expected and not a limitation; (ii) gains are large when disagreement is high; (iii) the operating regime is checkable on a small calibration set before deployment.

5.3 Designed vs. random references

A key question is whether the contrastive gain is structural or a prompt-engineering artifact. We test RANDOM (unstructured noise with no semantic content) as a noisy reference. RANDOM yields a 1.5-point gain at 8B and a 5.9-point gain at R1-7B over base on AIME'24 at 32K, refuting the hypothesis that gains are purely from output compression or prompt specificity. The gap between RANDOM and designed prompts is on the order of 1 to 3 points and reflects the additional signal from structured perturbation. This result also addresses the concern that **CTD** requires careful prompt engineering: any sufficiently different context provides useful signal.

Table 7: Random noise ablation (AIME'24, 32K). Designed prompts outperform random; both outperform base.

Model	Base	Rand.	Null	Opp.
Qwen3-8B	75.4	76.9	**77.7**	77.5
R1-Distill 7B	48.5	54.4	**57.7**	55.4

5.4 Statistical significance across configurations

Across 30 configurations (5 models $\times$ 3 benchmarks $\times$ 2 noise types), 24/30 show positive gains (sign test p=0.0002). On the independently powered GPQA Diamond, **CTD** shows a statistically significant gain at both 8B and 32B (McNemar p<0.001, Table 4). Temperature robustness: **CTD** Pass@1 varies negligibly across the temperature sweep we run (Appendix F), confirming the gain is structural rather than a sampling artifact. On MMLU, a clear majority of subjects win or tie under **CTD**, establishing no regression on general knowledge.

5.5 Why CTD outperforms both base and oracle

The oracle-copy gap (§2.2) shows Base > Oracle for R1-7B: the answer phase sometimes corrects trace errors. Standard decoding allows uncontrolled deviation, sometimes helpful (correcting trace errors) and sometimes harmful (overriding correct traces). The model has no mechanism to distinguish "I should deviate because the trace was wrong" from "I should deviate even though the trace was right." **CTD** provides that mechanism: the contrastive signal is proportional to the divergence between primary and noisy logits, which is larger when the trace carries strong instance-specific reasoning and smaller when trace confidence is low. This is consistent with the per-instance pattern in Table 6, though we do not directly measure trace confidence at the divergence token.

This explains the ordering CTD > Base > Oracle:

- **Oracle** forces strict trace-copying, suppressing the answer phase's corrective ability $\rightarrow$ lowest accuracy.
- **Base** allows uncontrolled deviation, net beneficial but noisy.
- **CTD** steers deviation toward corrections while suppressing errors $\rightarrow$ highest accuracy.

This property (contrastive steering rather than hard copying) is central to why **CTD** works and should inform future methods targeting the thinking-to-answer transition.

6 Discussion and conclusion

Summary. We studied the thinking-to-answer transition in large reasoning models and identified a pervasive phenomenon: final answers diverge from reasoning traces at rates between 6% and 48% across the LRMs we evaluate, even after conservative filtering for truncation artifacts and intermediate-answer aliasing. The oracle-copy gap, where naïve trace-copying underperforms standard decoding, reveals that the answer phase has independent corrective capacity, a nuance any steering method must respect. Contrastive Thinking Decoding (**CTD**) addresses this through contrastive logit reweighting against a noisy reference trace, concentrating CTD-internal joint mass on the both-correct cell

(T✓A✓) while keeping the CTD-internal drift cell (T✓A×) at or below 1.5% across the per-instance-logged conditions (Base also shows low drift in the same decomposition; the contrast is the joint shift, not a drift gap), and improving the accuracy–compute trade-off at every measured operating point on AIME and code while preserving it on near-saturated MATH500, all without parameter updates.

Trace–answer agreement as a separable control axis. Eliciting better reasoning and committing to a trace-consistent answer are separable objectives. This separation is not merely a property of the `<think>-<answer>` format: any autoregressive model must transition from derivation to conclusion, and at that transition point, generic language priors can overpower reasoning signals. Our results motivate treating trace-consistent answer commitment as a decoding-time control problem orthogonal to trace optimization.

What CTD does not claim. We do not claim that trace–answer disagreement is the dominant failure mode of reasoning models, nor that **CTD** resolves it completely. Disagreement rates vary across models and tasks (6% to 48%), and **CTD**'s gains are monotone in the disagreement rate (Spearman ρ=1.0 across five models on AIME'24 at 32K). On models with low disagreement (e.g., Qwen3-8B on AIME), gains are correspondingly modest. The practical value of **CTD** is highest for settings where trace–answer disagreement is high, a regime that naturally arises with smaller models, harder tasks, or budget-constrained inference. We do not interpret **CTD** as a faithfulness, alignment, or safety method (see Appendix A for the full scope statement).

Future directions. Promising extensions include instance-adaptive reference selection, compute-optimal composition with self-consistency and best-of-N, learned controllers for budget and prompt, and using contrastive signals to improve training objectives. The oracle-copy gap also motivates studying the division of labor between thinking and answering as a fundamental property of reasoning models. Detailed limitations, broader-impact analysis, and an extended discussion (relation to prior work, the Pass@1/Pass@N trade-off, instance-conditional contrast, and using **CTD** as a training signal) are in Appendix A, Appendix B, and Appendix C; theoretical analysis is in Appendix G.

Acknowledgments and Disclosure of Funding

No additional funding or competing interests to disclose beyond the listed affiliations.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: Llm learns when to think. *arXiv preprint arXiv:2505.13379*, 2025.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik Sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah Pratt, Vivek Ramanujan, Jon Saad-Falcon, Jeffrey Li, Achal Dave, Alon Albalak,

Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishaal Shankar, Aaron Gokaslan, Mike A. Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alexandros G. Dimakis, and Ludwig Schmidt. Openthoughts: Data recipes for reasoning models, 2025. URL https://arxiv.org/abs/2506.04178.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. Lighteval: A lightweight framework for llm evaluation, 2023. URL https://github.com/huggingface/lighteval.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Taehyeon Kim, Joonkee Kim, Gihun Lee, and Se-Young Yun. Instructive decoding: Instruction-tuned large language models are self-refiner from noisy instructions. *arXiv preprint arXiv:2311.00233*, 2023.

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*, 2022.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. Dexperts: Decoding-time controlled text generation with experts and anti-experts. *arXiv preprint arXiv:2105.03023*, 2021.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*, 2025.

MAA. American invitational mathematics examination (aime). https://www.maa.org/math-competitions/aime, 2024. Accessed: 2025-09-18.

MAA. American invitational mathematics examination (aime). https://www.maa.org/math-competitions/aime, 2025. Accessed: 2025-09-18.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Sean O'Brien and Mike Lewis. Contrastive decoding improves reasoning in large language models. *arXiv preprint arXiv:2309.09117*, 2023.

OpenAI. Gpt-5 system card. System card, OpenAI, August 2025. URL https://cdn.openai.com/gpt-5-system-card.pdf.

Yuxiao Qu, Matthew YR Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. Optimizing test-time compute via meta reinforcement fine-tuning. *arXiv preprint arXiv:2503.07572*, 2025.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2024.

LG Research, Kyunghoon Bae, Eunbi Choi, Kibong Choi, Stanley Jungkyu Choi, Yemuk Choi, Kyubeen Han, Seokhee Hong, Junwon Hwang, Taewan Hwang, et al. Exaone 4.0: Unified large language models integrating non-reasoning and reasoning modes. *arXiv preprint arXiv:2507.11407*, 2025.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R Johnston, et al. Towards understanding sycophancy in language models. *arXiv preprint arXiv:2310.13548*, 2024.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 783–791, 2024.

Vaishnavi Shrivastava, Ahmed Awadallah, Vidhisha Balachandran, Shivam Garg, Harkirat Behl, and Dimitris Papailiopoulos. Sample more to think less: Group filtered policy optimization for concise reasoning. *arXiv preprint arXiv:2508.09726*, 2025.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36, 2024.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.

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

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

Han Wu, Yuxuan Yao, Shuqi Liu, Zehua Liu, Xiaojin Fu, Xiongwei Han, Xing Li, Hui-Ling Zhen, Tao Zhong, and Mingxuan Yuan. Unlocking efficient long-to-short llm reasoning with model merging. *arXiv preprint arXiv:2503.20641*, 2025.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adaptthink: Reasoning models can learn when to think. *arXiv preprint arXiv:2505.13417*, 2025a.

Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, et al. A survey of reinforcement learning for large reasoning models. *arXiv preprint arXiv:2509.08827*, 2025b.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

Contents

1 Introduction 1

2 How answers diverge from reasoning traces 3
- 2.1 Measuring trace-answer disagreement . . . 3
- 2.2 The oracle-copy gap: answer-phase error correction . . . 4
- 2.3 Budget control alone is insufficient . . . 4

3 Contrastive thinking decoding 5

4 Experiments 5
- 4.1 Experimental setup . . . 5
- 4.2 Reducing trace-answer disagreement . . . 6
- 4.3 Accuracy–compute Pareto frontiers . . . 6
- 4.4 Generalization across tasks and scales . . . 7
- 4.5 Instance-level gains and decoding baselines . . . 7
- 4.6 Robustness and ablations . . . 8

5 Analysis 8
- 5.1 Asymmetric recovery vs. drift under **CTD** . . . 8
- 5.2 Gains scale predictably with disagreement . . . 8
- 5.3 Designed vs. random references . . . 9
- 5.4 Statistical significance across configurations . . . 9
- 5.5 Why CTD outperforms both base and oracle . . . 9

6 Discussion and conclusion 9

A Limitations 16

B Broader impact 17
- B.1 Positive impacts . . . 17
- B.2 Negative impacts and risks . . . 18
- B.3 Mitigations and recommendations . . . 19

C Discussion and future work 19
- C.1 Where the gain comes from, and where it does not . . . 19
- C.2 Trace–answer disagreement is a probabilistic disconnect, not a formatting artifact . . . 19
- C.3 Consistency: trading Pass@N for Pass@1 . . . 19
- C.4 Future direction: instance-conditional contrast . . . 20
- C.5 Future direction: prompt optimization and joint budget allocation . . . 20
- C.6 Future direction: from decoding rule to training signal . . . 20
- C.7 Relation to prior work, and what **CTD** does differently . . . 20

D AI usage 21

E Related work 21

F Extended experimental results 21

F.1 OpenThinker3 7B . . . 21
F.2 Unconstrained AIME accuracy across smaller models . . . 22
F.3 Qwen3-32B scalability table (max budget) . . . 22
F.4 Instance-level win rates . . . 22
F.5 Restricted disagreement on clean traces . . . 23
F.6 Stochastic thinking versus greedy thinking . . . 23
F.7 Ablation on λ . . . 24
F.8 Beyond math: sycophancy and general knowledge . . . 24
F.9 Temperature robustness . . . 24
F.10 Top-k masking ablation . . . 24
F.11 Consistency: Pass@1 vs. Pass@N trade-off . . . 24
F.12 Training-based method comparison . . . 25
F.13 Full budget sweep results . . . 25
F.14 LiveCodeBench full results . . . 25
F.15 Thinking prompt optimization . . . 27
F.16 Trace–answer disagreement as a probabilistic disconnect . . . 27

G Theoretical analysis 27

G.1 A decoding view of answer-phase agreement . . . 27
G.1.1 **CTD** is an entropy-regularized optimizer . . . 27
G.1.2 Under reference separation, off-trace tokens receive less probability . . . 28
G.2 Diagnostics for the separation assumption . . . 28
G.3 Supporting results . . . 29
G.4 Why NULL/OPPOSITE are effective noisy references . . . 29
G.5 Intuitive mechanism: separating reasoning from language priors . . . 29

H Experimental details 29

H.1 Thinking prompt templates . . . 29
H.2 Cutoff policy for budgeted inference . . . 30
H.3 Compute and latency analysis . . . 30
H.4 Boxed answer extraction . . . 31
H.5 Optional margin guard . . . 31

I Qualitative examples 32

I.1 AIME'25 Question 1: Geometry . . . 32
I.2 AIME'25 Question 4: Thinking-answer disagreement example . . . 32
I.3 Sycophancy: a non-math trace–answer split . . . 33
I.4 GPQA Diamond: science-reasoning recovery . . . 33
I.5 Side-by-side recoveries: Base vs. **CTD** . . . 34

A Limitations

Scope. Our primary evaluation centers on tasks with verifiable boxed answers: competition mathematics (AIME'24/'25, MATH500), multiple-choice science (GPQA Diamond), and code (LiveCodeBench). Beyond these, we report sycophancy on Sharma et al. [2024] and a broad-knowledge probe on MMLU (Table 4) as evidence that the trace–answer disagreement phenomenon is not an artifact of boxed-answer formatting; we do not extend the evaluation to long-form generation, multi-turn dialogue, tool use, or instruction-following at scale. The disagreement metric in Section 2 requires a parseable trace–answer separation, which transfers cleanly to two-phase `<think>`/`<answer>` models but not directly to open-ended outputs. All evaluated LRMs are open-weight distilled or RL-finetuned reasoning models in the 1.2B–32B range; we do not have access to end-to-end RL-trained frontier closed models, and disagreement rates may be quantitatively different there. Readers should treat the math/code/MCQ numbers as the regime where every claim is measured, and the sycophancy/MMLU numbers as a probe (not a deployment-ready guarantee) that the underlying disconnect is not format-specific.

Per-instance logging and decomposition coverage. The four-way decomposition into drift, recovery, persistent-correct, and persistent-wrong cells (Table 6, Table 3, Figure 5) requires per-instance answer logs aligned across the Base and **CTD** runs. Per-instance four-way logging is reported on three LRMs at the default λ=0.15: Exaone4 1.2B, R1-Distill 7B, and Qwen3-8B. Aggregate ΔAcc numbers in Table 18 cover a wider model pool, but the recovery-to-drift ratio of $\sim$13.6:1 and the per-instance drift bound of $\leq 1.5\%$ are properties of the three-LRM, default-λ subset. Smaller LRMs run at λ=0.4 contribute to aggregate accuracy but exhibit higher per-instance drift (up to 4.0% on individual benchmarks); we do not claim the bound generalizes uniformly across models or hyperparameters. Downstream readers should read the asymmetry claim as evidence of *direction* (recovery dominates drift in the default regime), not as a worst-case guarantee at arbitrary λ or model scale.

Budgeted inference heuristic. Our compute–accuracy trade-offs (Figure 4, Table 18) are obtained by closing the `<think>` block at the next sentence boundary after a token budget is reached. This rule is heuristic, model-dependent, and not learned: some models continue implicit reasoning into the answer span when the trace is cut early, others terminate cleanly, and the stable mid-range of budgets varies across tasks. The recommended $\lambda \in [0.05, 0.30]$ window in Appendix F is itself measured under this cutoff rule; a different cutoff (a learned controller, a verifier-gated stop, an EoS-only protocol) would shift the achievable frontier and could alter the optimal λ. We did not study a learned cutoff. For downstream readers, this means the absolute budget points reported here should be reproduced under the same cutoff rule before being interpreted as model-intrinsic.

Decoding hyperparameters and robustness. **CTD** introduces two decoding-time hyperparameters: the contrast strength λ and the noisy-prompt template (NULL by default; OPPOSITE as an alternative). Gains are stable across $\lambda \in [0.05, 0.30]$ for the three logged LRMs (Table 13); outside this range, behavior is not monotone: at large λ, Pass@1 can rise while Pass@N falls, consistent with a sharper but less diverse answer distribution. Each answer token requires an extra forward pass on the noisy-trace continuation, partially offset by **CTD** generating fewer total tokens than Base in our runs (Figure 6); the offset depends on the answer-phase fraction and is not guaranteed at large batch sizes that push the system into a compute-bound regime. The method is also brittle when the primary and noisy traces are highly correlated: the contrast term then approaches zero and **CTD** collapses to standard decoding. We use fixed λ and a fixed noisy template across runs and do not perform per-instance tuning. A practitioner reading this paper should expect the reported gains to require, at minimum, validation of λ on a held-out set within the recommended range, and should not deploy at extreme λ on the basis of Pass@1 alone.

Evaluation methodology. We summarize answer dynamics through accuracy versus decoded-token trade-offs (Pass@1, Pass@N, total tokens). This is a deliberate proxy and does not measure probability calibration, abstention quality, selective prediction, or downstream task utility. We do not report ECE, Brier scores, refusal curves, or human-judged answer quality. The Pass@1-vs-Pass@N tension noted above (Table 18, Table 19) is consistent with **CTD** trading sample diversity for adherence to the trace; whether this trade is desirable is task-dependent and we do not adjudicate it. The qualitative examples in Appendix I are illustrative and were not crowd-evaluated. Downstream

readers building on **CTD** for applications where calibration or diversity matters (active learning, exploration during RL training, ensemble verification) should re-evaluate under metrics appropriate to that setting; our numbers do not transfer.

Statistical power on small benchmarks. The within-pipeline McNemar tests reported in Table 20 are structurally limited on AIME'24 (n=30): a few-percent paired effect is below detection at α=0.05 regardless of consistency, and the AIME entries should be read as descriptive rather than confirmatory. The load-bearing significance result in our paper is GPQA Diamond (n=198), where paired effects reach $p<0.001$ at both 8B and 32B (Table 4). MATH500 is near-saturated for several models in our pool, which compresses both the disagreement range (sub-2% versus 6–48% on AIME-class benchmarks) and the room for ΔAcc gains; we do not interpret near-zero MATH500 deltas as evidence against **CTD**. For downstream readers, this means the GPQA, LiveCodeBench, and aggregate AIME-pool numbers carry the inferential weight; per-benchmark AIME comparisons are reported for completeness, not as standalone hypothesis tests.

What is not a CTD claim. We measure trace–answer *agreement*: whether the committal answer span matches the conclusion expressed at the end of the `<think>` block, under a protocol that holds the trace fixed and varies only the answer-phase decoder. This is a behavioral measurement and is distinct from chain-of-thought *faithfulness* in the sense of Turpin et al. [2024] and Lanham et al. [2023]: we do not establish that the trace is causally responsible for the model's internal computation, that the trace contains the actual reasoning the model performs, or that intervening on trace content would change the answer in a faithful way. The trace-held-fixed protocol is an experimental setup, not a causal-faithfulness claim. We also do not claim that **CTD** aligns reasoning models in any normative sense, mitigates safety failures, or generalizes the sycophancy result in Table 4 to a deployment-grade safety property: the $+1.6$ to $+2.4$ point absolute deltas there are evidence that the trace–answer mechanism extends beyond boxed-answer tasks, not that **CTD** is a sycophancy mitigation. Readers should treat **CTD** as a decoding-time method that reduces a measured behavioral disagreement, with no further entailment.

Pipeline reproducibility and within- vs. cross-pipeline numbers. All reported ΔAcc and ΔTokens are computed within a single inference pipeline (vLLM 0.10.1, fixed seeds, fixed sampling parameters, matched stop conditions for Base and **CTD**) so that the pairwise comparison isolates the decoder. Absolute numbers are not directly comparable to vendor-reported leaderboard scores or to numbers obtained under different sampling temperatures, prompt formats, system messages, or thinking-block templates. The thinking-prompt ablation in Section C.5 shows the noisy-trace template materially affects the contrast signal; NULL is our default and OPPOSITE is reported as an alternative, but other templates would yield different operating points. Readers reproducing our results should match the pipeline; readers comparing **CTD** to a different decoder (e.g., classical contrastive decoding, instructive decoding) should re-run both methods inside the same pipeline rather than cite our numbers against numbers reported elsewhere.

B Broader impact

CTD is a training-free, test-time decoding rule applied on top of publicly released open-source long-reasoning models (Qwen3, DeepSeek-R1-Distill, Exaone, OpenThinker3). It introduces no new model weights and no new datasets; the risk surface is therefore the *decoding-time behavior* that **CTD** induces, not the release of new capability. We discuss positive impacts, negative impacts and risks, and concrete mitigations below. We deliberately separate *what we measured* from *what we claim*: several effects below appear in our tables but should not be marketed as guarantees.

B.1 Positive impacts

Deployment cost and energy. **CTD** generates 240–1,270 fewer total tokens than Base across our studied configurations, and end-to-end mean latency on a single Nvidia H200 is 97.1% of Base for Qwen3-8B and 90.5% for R1-Distill 7B over sixteen runs on AIME'24 (Appendix H). At fixed accuracy, fewer tokens translate directly into lower energy and lower wall-clock cost per query. Because the method is training-free, the savings are realized without an additional training run, which itself carries a non-trivial energy footprint.

Drift rate as a deployment monitor. The drift-versus-recovery decomposition (§5.1) is itself a deployment artifact, not only an analysis tool. A serving system that runs both the Base trace and the **CTD**-aligned answer can log instance-level drift events and surface them as a real-time quality signal. Operators can flag deployments whose drift rate exceeds the $\sim 1.5\%$ ceiling we observed across audited LRMs at the default λ setting, treating excess drift as evidence of distribution shift, prompt regression, or contamination. This use of the metric is suggested, not validated; we have not run a multi-month deployment study.

Reproducibility and accessibility. **CTD** requires no parameter updates, no auxiliary model, and no proprietary tooling. A small research group or an independent auditor can reproduce every result with the open weights cited in §4.1 and a single H200 (or equivalent). This lowers the barrier to independent verification and to follow-up audits of LRM behavior.

Measured reduction in persona-yielding behavior. On the sycophancy probe in Table 4, Qwen3-8B improves from 30.4% to 32.0% and Qwen3-32B from 36.4% to 38.8% under **CTD**. We report this as evidence that the answer phase commits to the trace's conclusion under persona-pressure framing; we do not claim **CTD** is a safety method, an alignment method, or a sycophancy mitigation in any policy-relevant sense. The probe is a single benchmark in a narrow setting, and the gain is small in absolute terms.

B.2 Negative impacts and risks

Error amplification on flawed traces. The same mechanism that aligns the answer with the trace also locks in an incorrect trace. When the trace contains a faulty derivation, **CTD** suppresses precisely the off-trace continuations that would otherwise let the model recover by chance. Our recovery-versus-drift analysis quantifies this directly: the asymmetry favors recovery (ratio $\sim$13.6:1 in the audited regime), but drift events still occur and are concentrated on long, partially resolved traces. In domains where traces are systematically biased (e.g., culturally specific prompts, adversarial inputs, or domains far from the trace-curation distribution), this asymmetry could invert; we do not have evidence either way for those regimes.

Reduced output diversity. Pass@1 increases under **CTD** while Pass@N can decrease at the same operating point (Section C.3). For end-user serving, this is a feature: users get consistent, trace-aligned answers. For data-generation pipelines (synthetic-data fine-tuning, RL using rejection sampling, red-teaming), the reduction in tail diversity is a liability. Pipelines that assume i.i.d. samples from the Base distribution will see narrower coverage if **CTD** is silently inserted into the decoding stack.

Prompt-engineering misuse. **CTD** uses thinking-prompt variants such as NULL and OPPOSITE as the noisy reference. The same prompt-engineering surface can in principle be inverted: an operator could choose a positively phrased reference designed to push the answer toward a preferred outcome rather than away from a generic prior. Our framework does not prevent this; downstream reviewers should treat the choice of contrast prompt as a configuration parameter with policy implications, not a fixed implementation detail.

Hardware-class evaluation gap. Latency and energy claims here are measured on a single H200 with vLLM 0.10.1 at batch size one. Behavior under multi-tenant batching, on accelerator classes with different memory-bandwidth characteristics (mobile, edge, older datacenter GPUs), or on CPU-bound serving stacks may differ. The doubled answer-phase forward pass is a real cost; in compute-bound regimes the parallel-pass trick that yields our near-zero overhead may not hold.

Over-claim risk in downstream reading. The sycophancy-direction result, the persona-robustness reading, and the reduced-drift framing are all *measurements* of behavior on specific benchmarks. A common failure mode in downstream reading is to translate such measurements into safety claims ("**CTD** makes models more honest", "**CTD** reduces hallucination"). These translations are not supported by our experiments. We urge readers, reviewers, and downstream users to preserve the measurement-versus-claim distinction.

B.3 Mitigations and recommendations

We recommend that any deployment or downstream replication of **CTD** adopt the following practices: (i) report *both* accuracy and drift rate on every benchmark, with the drift rate disaggregated per model rather than pooled; (ii) audit trace-incorrect cases explicitly and report the rate at which **CTD** preserves wrong-trace conclusions, since this is the regime where error amplification dominates; (iii) disclose matched-compute footprints (tokens, FLOPs, wall-clock) for both Base and **CTD** configurations rather than only relative deltas; (iv) if used in a data-generation pipeline, mix in a fraction of non-contrastive samples or raise temperature in the thinking block to recover Pass@N; (v) maintain a clear separation between behavioral measurements (which our paper provides) and safety, alignment, or honesty claims (which our paper does not make and does not support). Future work should extend evaluation beyond math and code, study human factors such as user trust in trace-aligned answers, and assess energy and latency across hardware classes.

C Discussion and future work

C.1 Where the gain comes from, and where it does not

CTD is a decoding-time intervention that addresses one specific failure mode: the answer phase committing to a token sequence that is not entailed by an adequate thinking trace. On the controlled held-trace experiment of §2.2, the committed-answer change is the only degree of freedom, and the recovery cell of Table 3 outnumbers the drift cell by an order of magnitude. The aggregate Pareto picture in Figure 4 is therefore a plausible attribution: most of the headline gain comes from converting near-misses where the trace was already correct, not from improving traces that were wrong to begin with. This bounds what **CTD** cannot do: it does not repair a confidently wrong trace (the soft-anchoring asymmetry of §G.1), it does not extend the model's reasoning capacity, and it does not substitute for thinking-time compute when the trace is below threshold. Test-time control of LRM answer behavior is a broader research program; **CTD** is one tractable, training-free intervention inside that program, not a unified solution.

C.2 Trace–answer disagreement is a probabilistic disconnect, not a formatting artifact

A natural objection is that the disagreement we measure is a property of the explicit `<think>`/`<answer>` two-phase decoder rather than of reasoning behavior itself. We argue the opposite: the phenomenon is a probabilistic disconnect inherent to autoregressive generation. Every reasoning model, whether two-phase or not, must eventually transition from derivation to commitment, and at that transition two distributions compete: the reasoning signal accumulated in context, and the language prior over generic completions. Disagreement is the regime where the language prior overpowers the reasoning signal at the moment of conclusion. Under this view, **CTD**'s contrast against a noisy reference (which approximates the language prior; see §G.5) is a high-pass filter: it cancels the model's generic habits and preserves the trace-anchored signal. A consequence is that the same intervention should apply to no-summary models, although end-to-end RL-trained frontier models with mixed-format outputs remain to be tested.

C.3 Consistency: trading Pass@N for Pass@1

Across Table 18 and Table 19, we see that Pass@1 often increases under **CTD** while Pass@N can decrease at the same operating point. This pattern is consistent with a sharper answer distribution that adheres more tightly to the trace: **CTD** reduces variance from off-trace continuations and seed-level noise, producing fewer lucky hits but also fewer flips away from correct trace conclusions. The trade is desirable in deployments that prefer consistent, trace-aligned answers, and undesirable when the goal is exploration for data generation during training. Three mitigations preserve diversity when needed. First, raise temperature in the thinking block while keeping the answer-phase greedy, restoring diversity in the trace where it can be checked. Second, schedule the contrast weight λ_t so the contrast is strong only when the trace shows clear internal evidence (e.g., a final boxed value plus a verification step). Third, mix a small share of non-contrastive samples into a Maj@N pool to recover most of the Pass@N headroom while preserving the Pass@1 gain. A systematic study of these schedules is left to future work.

C.4 Future direction: instance-conditional contrast

The current implementation fixes λ globally and selects the noisy reference (NULL vs. OPPOSITE) at the family level: math defaults to NULL, code to OPPOSITE. The ablation in Table 13 shows a stable plateau over $\lambda \in [0.05, 0.30]$, which is what makes the global setting safe in practice. Adaptive variants are nonetheless attractive. A small predictor conditioned on inexpensive instance features (input length, a quick early-trace stub, or a verifier-style score over the trace) could pick λ and the prompt template per instance. Two natural objectives are accuracy at fixed compute and the recovery-minus-drift gap from Table 3. Cast as black-box optimization with bandit feedback over a small discrete prompt space, this is tractable on the same evaluation infrastructure used here. A learned verifier scoring the trace's internal evidence (§C.3) could in addition gate when contrast is applied at all, suppressing the Pass@N trade in trace-incomplete regimes.

C.5 Future direction: prompt optimization and joint budget allocation

We treated the thinking prompt as a fixed handle. The effects are measurable but coarse: EMPTY allows reasoning to spill into the answer, NULL blocks it, and OPPOSITE structures the trace at the cost of variance across model families. A natural next step is to optimize the prompt jointly with the thinking budget under explicit objectives (e.g., accuracy at fixed total compute, or stability under paraphrase). Useful structure here includes a small prompt composer that selects cues such as brevity or planning and emits a budget cap, trained from inexpensive instance features and reward signals from the recovery-vs-drift split. Robustness across paraphrase and adversarial cues inside the thinking block is the central evaluation axis: a practical protocol would report the accuracy–compute frontier together with drift rates over a held-out prompt set, mirroring Figure 4. Better prompts and **CTD** are complementary: prompts shape the trace, **CTD** anchors the answer to that trace at test time without parameter updates.

C.6 Future direction: from decoding rule to training signal

CTD is currently a test-time decoding modification. The contrast it computes (a per-token margin between the primary and noisy distributions) is also a free supervision signal. Two uses suggest themselves. First, distillation: pairs of (trace, **CTD**-aligned answer) are trace-faithful targets that can be used as preference data for DPO/IPO-style fine-tuning, internalizing the trace-conditioned answer commitment without the run-time double pass. Second, on-policy use: the **CTD** margin can serve as a token-level reward in RL fine-tuning to penalize answer-phase departures from the trace. Both directions move the trace-anchoring cost from inference to training, which is the preferred trade for high-throughput deployment. A third, more speculative direction is to use **CTD** as a data-generation rule: ablating the noisy reference recovers a baseline trajectory, and the difference between contrastive and Base answers identifies items where the language prior dominates, which is useful as a curriculum signal for further training.

C.7 Relation to prior work, and what CTD does differently

Three lines of work motivate the comparison set in Table 5. Classical contrastive decoding [Li et al., 2022, O'Brien and Lewis, 2023, Liu et al., 2021] pairs a strong model with a weaker auxiliary; **CTD** uses a single model with two thinking trajectories, removing the cross-model calibration problem and grounding the contrast in the model's own internal computation. Instructive decoding and context-aware decoding [Kim et al., 2023, Shi et al., 2024] steer generation through external instructions or added context; **CTD** uses the model's own thinking phase as the source of contrast and applies the rule only at test time. Self-consistency [Wang et al., 2022] draws N independent samples and aggregates their final answers; **CTD** operates at single-sample inference cost and is orthogonal in the sense that one can compute **CTD** inside each sample of a Maj@N pool. A direct head-to-head against self-consistency at matched compute (specifically, **CTD** at single-sample cost vs. Base at Maj@N with N matched in FLOPs) is a deployment-relevant baseline we did not include in this draft and treat as a near-term follow-up.

D AI usage

We use AI tools (OpenAI's ChatGPT, Google's Gemini, Anthropic's Claude including Claude Code, and LG AI Research's ChatExaone) to support code generation for experiments and to refine the manuscript at the sentence level. The core idea of the method, the design of the framework, the experimental protocol, and the substantive content of the manuscript are written by the authors. AI-assisted edits to text and code were reviewed and verified by the authors before inclusion. No AI tool was used as a non-standard component of the methodology development.

E Related work

Reasoning in LLMs. Chain-of-thought prompting [Wei et al., 2022, Kojima et al., 2022] and its extensions [Zhou et al., 2022, Chen et al., 2022] elicit reasoning at inference time. Large reasoning models (LRMs) internalize this via RL-optimized thinking phases [Jaech et al., 2024, Guo et al., 2025], with recent work optimizing trace quality [Shrivastava et al., 2025, Zhang et al., 2025b, Zhao et al., 2025, Wang et al., 2025]. Our work complements trace optimization by focusing on the under-explored answer phase.

Test-time compute scaling. Budget forcing [Muennighoff et al., 2025], adaptive computation methods [Zhang et al., 2025a, Fang et al., 2025], optimal-compute allocation [Snell et al., 2025], and training-based efficiency methods [Chen et al., 2024, Luo et al., 2025] control *how much* the model thinks; repeated-sampling and verifier-rerank methods [Wang et al., 2022, Lightman et al., 2023] control *how many* candidates are scored. **CTD** addresses a complementary axis: *how faithfully the answer reflects the thinking trace it conditions on*. These approaches compose naturally; **CTD** shifts the Pareto frontier upward at every budget level.

Contrastive and guided decoding. Contrastive decoding [Li et al., 2022] pairs a strong model with a weaker auxiliary model. DExperts [Liu et al., 2021] and context-aware decoding [Shi et al., 2024] use external expert/anti-expert models or context perturbations. Instructive decoding [Kim et al., 2023] injects instructions as control cues. **CTD** differs by operating within a single model, using the thinking trace itself as the source of contrast rather than relying on auxiliary models or external instructions.

Faithfulness and self-consistency. Self-consistency [Wang et al., 2022] samples multiple reasoning paths and votes on the final answer, addressing stochastic variation. Recent work on reasoning faithfulness studies whether chain-of-thought actually reflects model computation [Turpin et al., 2024, Lanham et al., 2023]. Our work identifies a related but distinct phenomenon: the answer phase can actively deviate from the trace even when the trace is deterministically fixed, and this deviation has both harmful (drift) and beneficial (error correction) components. The sycophancy literature [Sharma et al., 2024] shows that models can produce outputs that conflict with their internal reasoning under social pressure; we demonstrate that this same trace-answer split occurs naturally, without adversarial prompting, in the standard reasoning pipeline.

F Extended experimental results

F.1 OpenThinker3 7B

OpenThinker3 [Guha et al., 2025] uses an open data-recipe reasoning training pipeline distinct from the Qwen, DeepSeek-R1-Distill, and Exaone families that dominate the rest of our evaluation. Table 8 reports **CTD** applied to this LRM without any modification of the contrastive coefficient or thinking-prompt templates. On AIME'24 **CTD** improves Pass@1 by $+1.74$ (NULL) and $+0.70$ (OPPOSITE) over Base, and on AIME'25 by $+5.66$ and $+6.91$ respectively. MATH500 is at near-saturation (95.00% under Base) and **CTD** preserves it within $+0.4$ to $+0.6$ points. Token counts under **CTD** are within $\pm 3\%$ of Base at every operating point. The transfer establishes that the trace–answer disagreement phenomenon and **CTD**'s correction mechanism are not specific to a single model family or training recipe.

Table 8: **OpenThinker3 7B** at unconstrained thinking budget. **CTD** transfers to this architecture without modification.

Method	MATH500		AIME'24		AIME'25	
	Tokens	Pass@1 (%)	Tokens	Pass@1 (%)	Tokens	Pass@1 (%)
Base	4548	95.00	14888	65.76	15243	53.30
CTD (Null)	4648	95.40	14484	67.50	14843	58.96
CTD (Opposite)	4672	95.60	14788	66.46	14854	60.21

F.2 Unconstrained AIME accuracy across smaller models

We report unconstrained-budget AIME Pass@1 for the three smaller LRMs (Exaone4 1.2B, R1-Distill 1.5B, Qwen3 1.7B) in Table 9; entries are the better of NULL and OPPOSITE across 16 runs on n=30 problems each. **CTD** improves AIME'24 by +3.0 (Exaone4), +1.2 (R1-Distill 1.5B), and +3.3 (Qwen3 1.7B), and AIME'25 by +3.6, +3.5, and +4.0 respectively. The magnitude of the gain tracks per-LRM disagreement: smaller LRMs exhibit higher disagreement at full budget and receive correspondingly larger **CTD** gains, matching the monotonicity reported in Section 5.2 and the contraction-bound prediction.

Table 9: Unconstrained budget AIME Pass@1 (best of NULL/OPPOSITE, 16 runs, n=30).

	AIME'24		AIME'25	
Model	**Base**	**CTD**	**Base**	**CTD**
Exaone4 1.2B	56.0	**59.0**	42.9	**46.5**
R1-Distill 1.5B	33.8	**35.0**	22.3	**25.8**
Qwen3 1.7B	44.2	**47.5**	34.8	**38.8**

F.3 Qwen3-32B scalability table (max budget)

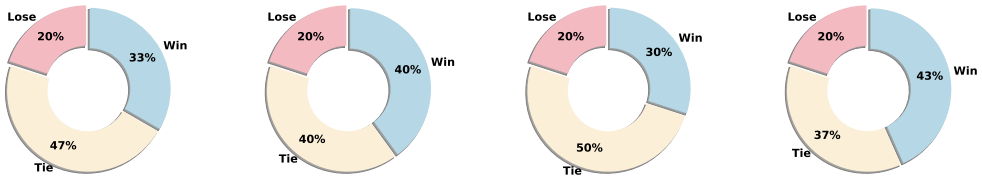
At 32B the AIME benchmarks approach near-saturation (Base 81.4% on AIME'24 and 72.9% on AIME'25; n=30 problems each, so a one-point delta corresponds to a single problem). Table 10 reports the same configuration under **CTD**: MATH500 is held at 97.2%, AIME'24 **CTD**-OPPOSITE reaches 82.9 (+1.5 points $\approx$ 0.5 of one problem on average), and AIME'25 is preserved within 0.3 points. We treat these AIME deltas as preservation rather than improvement evidence; the load-bearing 32B scaling result is GPQA Diamond at n=198 items, which yields +6.0 points at 8B and +4.5 points at 32B with McNemar $p<0.001$ in both cases (Table 4). The scalability narrative is therefore that **CTD**'s gain shrinks as accuracy approaches saturation but does not flip negative; on benchmarks with statistical headroom (GPQA Diamond) the gain remains significant at 32B.

Table 10: Qwen3-32B AIME and MATH500 at max thinking budget. AIME numbers should be read as preservation under **CTD** (n=30, fraction-of-one-problem deltas); the load-bearing 32B scaling evidence is GPQA Diamond at n=198 in Table 4.

Method	**MATH500**	**AIME'24**	**AIME'25**
Base	97.2	81.4	72.9
CTD (NULL)	97.2	81.3	**73.2**
CTD (OPPOSITE)	97.2	**82.9**	**73.1**

F.4 Instance-level win rates

Beyond aggregate accuracy, we decompose per-problem outcomes into four cases: both **CTD** and Base correct, only **CTD** correct (**CTD** win), only Base correct (Base win), and both wrong. Figure 5 shows that on Qwen3-8B AIME, **CTD**-win cases outnumber Base-win cases under both NULL and OPPOSITE variants and on both AIME'24 and AIME'25. The asymmetry indicates that **CTD**'s gain is not a wash from gain-loss cancellation but a structural per-problem shift toward the both-correct cell, consistent with the four-way decomposition in Table 3.



(a) NULL, '24. (b) NULL, '25. (c) OPP, '24. (d) OPP, '25.

Figure 5: **CTD** vs. Base instance-level win rates (Qwen3-8B AIME, 16 runs per problem).

F.5 Restricted disagreement on clean traces

Section 2.1 addresses truncation as a confound by restricting the disagreement metric to *clean* traces (terminate naturally + contain a `\boxed{}`). Table 11 reports the per-model audit underlying the 6.4%–48.2% band claim in the main text: even on the clean subset, disagreement on AIME spans 6.4% (Qwen3-8B) to 48.2% (Exaone4 1.2B), refuting the hypothesis that the headline range is a truncation artifact. Clean-trace coverage exceeds 91% at the 32K budget for every model except Qwen3-1.7B (already excised; see Table 21), so the restriction does not throw away meaningful sample size. The oracle column reports the accuracy obtained by taking the trace's last-boxed answer as the final answer; oracle accuracy is essentially identical to trace accuracy across rows, confirming that the disagreement metric is not an extraction-pipeline artifact.

Table 11: Restricted-disagreement audit on AIME at the 32K budget. Clean traces terminate naturally with a `\boxed{}`; clean disagreement is the fraction of clean traces whose answer-phase boxed answer differs from the trace boxed answer. Trace-acc and oracle-acc (using the trace's last-boxed answer as the final answer) coincide, confirming the disagreement metric is not extraction-pipeline noise.

Model	**Bench.**	**Var.**	N	**Clean (%)**	**Disagree (%)**	**Trace acc.**	**Oracle acc.**
R1-Distill 7B	AIME'24	Null	480	99.4	33.3	27.3	27.3
R1-Distill 7B	AIME'24	Opp.	480	99.8	35.5	23.6	23.6
R1-Distill 7B	AIME'25	Null	480	100.0	29.2	33.3	33.3
Qwen3-8B	AIME'24	Null	480	91.7	6.4	55.2	55.2
Qwen3-8B	AIME'24	Opp.	480	91.3	8.4	54.8	54.8
Qwen3-8B	AIME'25	Null	480	92.7	20.0	67.0	67.0
Exaone4 1.2B	AIME'24	Null	480	99.8	48.2	25.7	25.7
R1-Distill 1.5B	AIME'24	Null	480	98.5	21.4	20.9	20.9
R1-Distill 1.5B	AIME'25	Null	480	99.4	18.9	23.3	23.3

F.6 Stochastic thinking versus greedy thinking

Table 12: Stochastic vs. greedy allocation in thinking vs. answering. Accuracy (avg. pass@1/pass@4) (%) on MATH500 with R1-Distill 1.5B under varying thinking token budgets.

Method	Metrics	MATH500 (R1-Distill Qwen-1.5B)				
1 Greedy thinking & 4 Stochastic answering	Thinking budget Acc.	1k 77.20 / 87.80	2k 80.80 / 90.80	4k 81.60 / 91.20	8k 78.70 / 89.40	16k 79.80 / 90.20
4 Stochastic thinking & 1 Greedy answering	Tokens Acc.	1k 78.20 / 88.40	2k 82.40 / 90.40	4k 84.00 / 92.40	8k 84.20 / 92.40	16k 84.80 / 94.20

Accuracy when diversity is injected in the answer phase (1 greedy thinking, 4 stochastic answering) versus in the thinking phase (4 stochastic thinking, 1 greedy answering). Despite equal token budgets, only the latter improves accuracy, confirming that repeated answer-level sampling of the same path is ineffective, whereas distinct thinking paths yield ensemble gains.

Table 13: Ablation on λ for R1-Distill 7B at the unconstrained setting. A stable plateau exists around $\lambda \in [0.05, 0.30]$.

Dataset / λ	Method	**0.00**	**0.05**	**0.10**	**0.15**	**0.20**	**0.25**	**0.30**	**0.35**	**0.40**
AIME'24	Base	51.46	—	—	—	—	—	—	—	—
	CTD (Null)	51.46	55.13	56.66	57.71	56.92	56.66	57.71	53.46	49.16
	CTD (Opposite)	51.46	54.71	56.33	**56.88**	56.33	55.76	**56.88**	52.88	51.46
AIME'25	Base	40.20	—	—	—	—	—	—	—	—
	CTD (Null)	40.20	42.71	40.20	42.71	42.50	42.80	42.22	40.11	40.05
	CTD (Opposite)	40.20	42.50	40.46	42.50	42.22	42.67	42.12	39.91	40.05

F.7 Ablation on λ

We observe a small plateau rather than a monotone curve. On AIME'24, the best region includes λ near 0.15 and 0.30. Very large λ reduces accuracy, consistent with over-penalizing the noisy trace. We fix $\lambda = 0.15$ for all main experiments.

F.8 Beyond math: sycophancy and general knowledge

CTD extends to one non-math, no-`\boxed{}` setting (Sharma 2024 sycophancy, n=500, 16 runs). On items presented with a stated user preference for a wrong option, base outputs disagree with their own thinking traces in a non-trivial fraction of cases; **CTD** narrows that gap:

Table 14: Beyond-math evaluation.

Task	**Model**	**Base**	**CTD**
Sycophancy (n=500)	Qwen3-8B	30.4	32.0 (+1.6)
Sycophancy (n=500)	Qwen3-32B	36.4	38.8 (+2.4)
TruthfulQA MC1 (n=817)	R1-7B	75.8	76.0 (+0.2)
MMLU (1,531 Q, 58 subj.)	Qwen3-8B	82.8	83.1 (+0.3)

On MMLU, 51 of 58 subjects show wins or ties, confirming no regression on general knowledge.

F.9 Temperature robustness

Table 15: Temperature robustness of **CTD** Pass@1 (%). Qwen3-8B, AIME'24, budget=16K.

T=0.0	T=0.3	T=0.6	T=0.9	σ
69.8	70.2	69.4	69.6	0.3

The contrastive signal operates at the logit level before sampling, making the gain structural rather than a sampling artifact.

F.10 Top-k masking ablation

Table 16 examines whether the contrastive correction needs to apply over the full vocabulary at each answer-phase token. We restrict the contrastive logit subtraction to the top-k tokens (ranked by base-model probability) and re-evaluate on R1-Distill 1.5B at the unconstrained budget. At top-k=128, Pass@1 and Maj@N are essentially unchanged from top-k=∞: NULL 86.60 vs 86.60, OPPOSITE 86.20 vs 86.20. The contrastive signal localizes to the head of the next-token distribution, so top-k truncation does not cost accuracy and is a useful optimization where the vocabulary-size logit-difference computation is a memory or bandwidth bottleneck.

F.11 Consistency: Pass@1 vs. Pass@N trade-off

The Pass@1-vs-Pass@N trade-off and three concrete mitigations are discussed in §C.3.

Table 16: Top-k masking during answer-phase extrapolation (R1-Distill 1.5B, max budget).

Setting	Method	Tokens	Pass@1	Maj.
Top-k=∞	Base	3733	84.35	89.60
	CTD (Null)	3848	**86.60**	90.20
	CTD (Opp)	3997	86.20	**90.40**
Top-k=128	**CTD** (Null)	3848	**86.60**	90.00
	CTD (Opp)	3997	86.20	90.00

F.12 Training-based method comparison

Table 17 compares **CTD** against budget-aware training methods on R1-Distill 1.5B. The setup is two-fold: a tight token budget (Max-4K), where the comparison is against budget-shaping RL/FT methods (Thinkless, AdapThink, OverThink, O1-Pruner, DPO_shortest, Model Merging), and a full-budget setting against meta-RL (MRT). At Max-4K, **CTD** reaches 28.1 Pass@1 on AIME'24 versus Thinkless's 27.3 and 82.2 on MATH500 versus 81.8, while consuming fewer tokens (6,238 vs 7,099) and requiring zero parameter updates. At full budget, **CTD** reaches 35.0 Pass@1 on AIME'24 versus MRT's 30.3 and 86.6 on MATH500 versus 80.4. The takeaway is not that **CTD** dominates training-based methods on every axis: budget-RL methods still have an architectural advantage when one wants to learn problem-conditional budget allocation. The takeaway is that a training-free decoding-time method can be competitive on the budget-aware leaderboard without paying the training cost, and the two approaches are compositional—one can train a budget-shaping model and apply **CTD** at decoding time on top of it.

Table 17: Comparison with training-based methods on AIME '24 (DeepSeek-R1-Distill-1.5B).

			AIME '24		MATH500	
Method	**Type**	**Training?**	**Pass@1 (%)**	**Avg. Tokens**	**Pass@1 (%)**	**Avg. Tokens**
Base Model	-	No	17.3	10,615	79.0	3,000
Category 1: Budget-Aware Training Methods						
Model Merging [Wu et al., 2025]	Merge	Yes	17.3	10,615	-	-
Thinkless [Fang et al., 2025]	RL	Yes	27.3	7,099	81.8	2,555
OverThink [Chen et al., 2024]	FT	Yes	28.3	11,269	81.2	4,131
O1-Pruner [Luo et al., 2025]	FT	Yes	28.9	8,982	85.0	3,007
DPO_shortest [Rafailov et al., 2023]	RL	Yes	30.7	10,794	82.4	3,708
AdapThink [Zhang et al., 2025a]	RL	Yes	31.3	8,686	83.4	2,337
CTD (Null – Max 4K)	Decoding	**No**	**28.1**	**6,238**	**82.2**	**2,074**
Category 2: Meta-RL / Full Budget						
MRT [Qu et al., 2025]	Meta-RL	Yes	30.3	-	80.4	-
CTD (Null – Full Budget)	Decoding	**No**	**35.0**	12,431	**86.6**	4,608

F.13 Full budget sweep results

Table 18 consolidates the per-(model, budget, variant) Pass@1 numbers underlying the Pareto frontiers in Section 4.3. Three patterns are visible across the sweep. First, at the smallest budget (4K) on AIME, **CTD**-NULL can be conservative (matches or slightly underperforms Base) while OPPOSITE occasionally exceeds Base; this reflects that at small budgets much of the answer is doing extra reasoning that the trace cannot reliably anchor (the over-answering regime of Section 2.3). Second, at intermediate budgets (8K–16K) gains stabilize and exceed Base across both variants on AIME, while remaining within noise on near-saturated MATH500. Third, at the maximum budget (32K) gains peak: AIME'24 R1-Distill 7B reaches 57.71 (NULL) versus Base 51.46, and AIME'25 reaches 41.04 (NULL) versus Base 39.72. The token columns show that **CTD** typically generates fewer total tokens than Base at the same budget, partly offsetting the doubled answer-phase forward pass (compute analysis in Appendix H).

F.14 LiveCodeBench full results

On LiveCodeBench (Table 19), **CTD**-OPPOSITE dominates NULL across both LRMs and most operating points. At R1-Distill 7B 32K, OPPOSITE reaches Pass@1 43.75, Pass@5 54.96, Pass@10

Table 18: Full results on R1-Distill Qwen 7B and Qwen3 8B across all budgets and benchmarks. Arrows indicate change vs. Base at the same model and budget.

			Math500 (N=4)		**AIME'24** (N=16)		**AIME'25** (N=16)		**LiveCodeBench** (N=10)	
Model	**Budget**	**Method**	**Tokens**	**Pass@1**	**Tokens**	**Pass@1**	**Tokens**	**Pass@1**	**Tokens**	**Pass@1**
R1-Distill (7B)	4K	Base	2093	92.30	7798	49.58	7798	36.04	6557	35.83
		CTD (NULL)	1927	92.40	4366	44.79	4245	35.63	6557	35.83
		CTD (OPP)	1799	92.20	5299	50.41 ⇑	3988	35.63	6557	35.83
	8K	Base	2283	93.20	6466	49.48	6466	36.04	8063	39.17
		CTD (NULL)	2186	93.90 ⇑	6952	50.42 ⇑	5480	38.13 ⇑	8063	39.17
		CTD (OPP)	2150	93.75 ⇑	5577	50.41 ⇑	6089	39.38 ⇑	8341	40.42 ⇑
	16K	Base	2454	94.20	9650	50.00	8927	38.75	10580	41.67
		CTD (NULL)	2417	94.20	8093	**56.88** ⇑	9076	41.25 ⇑	10580	41.67
		CTD (OPP)	2509	**94.55** ⇑	8328	55.21 ⇑	9306	**42.50** ⇑	10580	41.67
	32K	Base	2685	94.30	10499	51.46	10545	40.20	11085	42.92
		CTD (NULL)	2509	**94.55** ⇑	8741	57.71 ⇑	10707	41.88 ⇑	11575	40.00
		CTD (OPP)	2487	94.50 ⇑	8492	55.41 ⇑	11398	42.71 ⇑	10820	**43.75** ⇑
Qwen3 (8B)	4K	Base	3118	92.95	7183	50.00	7415	42.71	13357	38.33
		CTD (NULL)	3157	93.50 ⇑	6943	50.83 ⇑	6212	41.46	13185	36.25
		CTD (OPP)	3157	93.50 ⇑	6777	63.54 ⇑	7087	42.91 ⇑	12361	40.42 ⇑
	8K	Base	3668	95.05	8239	62.71	9118	50.21	11943	43.33
		CTD (NULL)	3868	95.25 ⇑	8134	63.33 ⇑	9556	50.00	12529	45.83 ⇑
		CTD (OPP)	3841	95.10 ⇑	7788	63.54 ⇑	9893	50.21	12529	45.83 ⇑
	16K	Base	4178	95.85	11008	69.17	12702	58.54	13453	47.92
		CTD (NULL)	4419	96.40 ⇑	10448	72.29 ⇑	12462	61.88 ⇑	13453	47.92
		CTD (OPP)	4285	96.40 ⇑	10225	71.67 ⇑	12462	61.88 ⇑	13543	**50.83** ⇑
	32K	Base	4202	96.10	13701	75.00	15131	64.79	-	-
		CTD (NULL)	4608	96.85 ⇑	12431	77.71 ⇑	14405	67.71 ⇑	-	-
		CTD (OPP)	4706	**97.00** ⇑	13134	**77.50** ⇑	15294	**68.33** ⇑	-	-

58.33 versus Base 42.92 / 49.07 / 54.17, while NULL regresses to Pass@1 40.00; at Qwen3-8B 16K, OPPOSITE reaches Pass@1 50.83 versus Base 47.92. We attribute the variant gap on code to the structure of code answers: NULL collapses the trace prior aggressively, which is appropriate for math (the answer is a short numeric token), while OPPOSITE preserves a richer adversarial-but-coherent reference, leaving more head-room for the answer phase's program-construction component. We therefore recommend OPPOSITE as the default noisy reference for code while keeping NULL as the default for math. At low budgets (4K) on R1-Distill 7B, all three methods coincide because the answer phase rarely produces a different program when the trace is already short; the variant separation emerges at $\geq$ 8K where the answer phase has room to deviate.

Table 19: Full LiveCodeBench results (Pass@1/5/10). Arrows indicate change vs. Base.

Model	**Budget**	**Method**	**Tokens**	**Pass@1**	**Pass@5**	**Pass@10**
R1-Distill 7B	4K	Base	6557	35.83	45.39	50.00
		CTD (NULL)	6557	35.83	45.39	50.00
		CTD (OPP)	6557	35.83	45.39	50.00
	8K	Base	8063	39.17	51.95	54.17
		CTD (NULL)	8063	39.17	51.95	54.17
		CTD (OPP)	8341	40.42 ⇑	50.81	54.17
	16K	Base	10580	41.67	51.39	54.17
		CTD (NULL)	10580	41.67	51.39	54.17
		CTD (OPP)	10580	41.67	51.39	54.17
	32K	Base	11085	42.92	49.07	54.17
		CTD (NULL)	11575	40.00	47.92	50.00
		CTD (OPP)	10820	**43.75** ⇑	**54.96** ⇑	**58.33** ⇑
Qwen3 8B	4K	Base	13357	38.33	45.70	45.83
		CTD (NULL)	13185	36.25	46.88 ⇑	50.00 ⇑
		CTD (OPP)	12361	40.42 ⇑	48.97 ⇑	50.00 ⇑
	8K	Base	11943	43.33	54.05	58.33
		CTD (NULL)	12529	45.83 ⇑	52.31	54.17
		CTD (OPP)	12529	45.83 ⇑	52.31	54.17
	16K	Base	13453	47.92	**57.39**	**62.50**
		CTD (NULL)	13453	47.92	**57.39**	**62.50**
		CTD (OPP)	13543	**50.83** ⇑	54.07	54.17

F.15 Thinking prompt optimization

The thinking prompt is treated as a simple handle in this paper; a deeper discussion of prompt optimization, joint budget allocation, and how prompt tuning and **CTD** compose is in §C.5.

F.16 Trace–answer disagreement as a probabilistic disconnect

The argument that trace–answer disagreement reflects a probabilistic disconnect inherent to autoregressive generation, rather than the two-phase output format, is developed in §C.2.

G Theoretical analysis

G.1 A decoding view of answer-phase agreement

Why a theory section. Our experiments show that improving the thinking trace does not automatically yield a trace-consistent final answer. We therefore model **CTD** as a *test-time* objective that reweights answer-token probabilities using two traces generated by the *same* model: a *primary* trace that reflects the model's default reasoning under a fixed thinking budget, and a *noisy reference* trace induced by a designed thinking prompt (e.g., NULL or OPPOSITE). This section formalizes **CTD** as an entropy-penalized optimizer and states conditions under which it suppresses off-trace and redundant answer behavior.

Problem setup

Let p_θ be a reasoning model over vocabulary $\mathcal{V}$. Given input x, we generate a *primary* thinking trace $s^P_{1:T_P}$ with thinking budget B_{think}. Optionally, using a designed thinking prompt, we generate a *noisy reference* trace $s^N_{1:T_N}$ under its own reference budget B_N. In this section, we *do not allocate a separate budget* for the noisy reference. Instead, the reference trace is treated as additional *input-prefill tokens* and counted toward the total compute budget. We then decode the answer $y_{1:U}$ under answer budget B_{ans}. At answer step t, define next-token distributions

$$\pi^P_t(\cdot) = p_\theta\big(\cdot \mid x, s^P, y_{<t}\big), \qquad \pi^N_t(\cdot) = p_\theta\big(\cdot \mid x, s^N, y_{<t}\big) \ \text{(if } s^N \text{ exists)}.$$

An *aggregator* A maps (π^P_t, π^N_t) to a controlled distribution q_t (if s^N is absent, A depends only on π^P_t), and the decoder draws $y_t \sim \mathrm{Dec}(q_t)$.

Budget rule (simple stepwise cutoff). We use a fixed stopping rule for trace generation. The thinking phase stops when the token budget is exhausted or when the suffix matches a sentinel string `.\n\n`. The same rule applies to the noisy reference when used. This makes the overall compute predictable and leaves the answer-phase control problem to (A, Dec).

Lemma 1 (Budget feasibility). Let $C(x, s^N, y) = c_P T_P + c_N T_N + c_A U$ with $c_P, c_N, c_A > 0$. Under the stopping rule, for any (A, Dec),
$C(x, s^N, y) \le B_{\text{think}} + B_{\text{ans}}$ if s^N is not formed, $\quad C(x, s^N, y) \le B_{\text{think}} + B_{\text{ans}} + B_N$ if s^N uses budget B_N.
Proof. By construction, $c_P T_P \le B_{\text{think}}$, $c_A U \le B_{\text{ans}}$, and $c_N T_N \le B_N$ when applicable. □

Contrastive aggregator (CTD). Fix smoothing $\varepsilon > 0$ and schedules $\alpha_t \ge 1$ and $\lambda_t \ge 0$. Define

$$q_t(v) \ \propto \ \frac{\big(\pi^P_t(v)\big)^{\alpha_t}}{\big(\pi^N_t(v) + \varepsilon\big)^{\lambda_t}}, \quad v \in \mathcal{V}. \tag{1}$$

If s^N is absent, we use $q_t(v) \propto (\pi^P_t(v))^{\alpha_t}$.

G.1.1 CTD is an entropy-regularized optimizer

Proposition 1 (Variational characterization). For fixed π^P_t, π^N_t, α_t, λ_t, and ε, the distribution q_t in equation 1 is the unique maximizer over $r \in \Delta(\mathcal{V})$ of

$$\Phi_t(r) = \alpha_t \, \mathbb{E}_{v\sim r}\big[\log \pi^P_t(v)\big] - \lambda_t \, \mathbb{E}_{v\sim r}\big[\log\big(\pi^N_t(v) + \varepsilon\big)\big] + H(r),$$

where $H(r) = -\sum_v r(v) \log r(v)$.

Interpretation. **CTD** maximizes agreement with the primary trace while penalizing tokens preferred by the noisy reference, with entropy preventing degenerate collapse.

Proof. Φ_t is strictly concave in r. Stationarity yields $r(v) \propto (\pi_t^P(v))^{\alpha_t}/(\pi_t^N(v) + \varepsilon)^{\lambda_t}$, hence $r = q_t$; uniqueness follows. □

G.1.2 Under reference separation, off-trace tokens receive less probability

Assumption N1 (Separation of the noisy reference). Let $\mathcal{V} = S \cup U$ be a partition of tokens at step t into a *supported* set S and an *unsupported* set U. Intuitively, S contains tokens compatible with trace-anchored finalization from s^P, while U contains tokens that move the generation away from that finalization. Assume there exists $\delta > 0$ such that for all $s \in S$ and $u \in U$,

$$\frac{\pi_t^N(u) + \varepsilon}{\pi_t^N(s) + \varepsilon} \;\geq\; e^{\delta}.$$

Remark. Assumption N1 states that the noisy reference is systematically more permissive of unsupported tokens than of supported ones. This is a sufficient condition for **CTD** to suppress off-trace probability mass. A diagnostic proxy for δ can be constructed from logged π_t^N when such logging is available (see Section G.2).

Proposition 2 (Contraction of unsupported mass). Under Assumption N1 and any $\alpha_t \geq 1$, $\lambda_t > 0$,

$$q_t(U) \;:=\; \sum_{u \in U} q_t(u) \;\leq\; \frac{e^{-\lambda_t \delta}}{e^{-\lambda_t \delta} + \Theta_t}, \qquad \Theta_t \;:=\; \frac{\sum_{s \in S} (\pi_t^P(s))^{\alpha_t}}{\sum_{u \in U} (\pi_t^P(u))^{\alpha_t}}.$$

Interpretation. When the noisy reference favors unsupported tokens over supported ones, **CTD** reduces $q_t(U)$ exponentially in $\lambda_t \delta$, up to the primary signal-to-noise ratio Θ_t.

Proof. Apply Assumption N1 to bound $(\pi_t^N + \varepsilon)^{-\lambda_t}$ on S and U in equation 1, then convert the resulting odds bound into a probability bound. □

Connection to the oracle-copy gap. The soft anchoring property of **CTD** ($\lambda < 1$ in practice) connects directly to the oracle-copy gap (Section 2.2). When the primary trace contains an error, the primary distribution π_t^P may assign low confidence to trace-consistent tokens. In such cases, the contrastive correction $-\lambda \ell_t^N$ is insufficient to override the answer phase's independent computation, allowing corrective deviations to persist. Conversely, when the trace is correct and confident, the contrastive term reinforces trace-consistent tokens more strongly. This asymmetry explains why **CTD** achieves CTD $>$ Base $>$ Oracle: it preserves corrective deviations (which benefit from low trace confidence) while suppressing erroneous ones (which occur despite high trace confidence).

G.2 Diagnostics for the separation assumption

Proxy statistic. If one logs π_t^N during decoding, a simple proxy for the separation strength is

$$\widehat{\delta}_t = \mathrm{median}_{u \in U,\, s \in S} \log \frac{\pi_t^N(u) + \varepsilon}{\pi_t^N(s) + \varepsilon}.$$

Larger values of $\widehat{\delta}_t$ indicate stronger separation and, by Proposition 2, predict stronger contraction of unsupported mass.

Constructing (S, U) from trace anchors. A reproducible choice is to define anchors from s^P using verifiable artifacts such as a final boxed number, a selected option, or an extracted numeral span. One can then place tokens that extend or finalize the anchored answer into S, and treat the remaining tokens as U.

G.3 Supporting results

Pairwise odds tilt. For any tokens $a, b \in \mathcal{V}$,

$$\frac{q_t(a)}{q_t(b)} = \Big(\frac{\pi_t^P(a)}{\pi_t^P(b)}\Big)^{\alpha_t} \Big(\frac{\pi_t^N(a)+\varepsilon}{\pi_t^N(b)+\varepsilon}\Big)^{-\lambda_t}.$$

Thus, tokens that are more favored by the primary trace and less favored by the noisy reference gain odds under **CTD**.

Answer-phase elicited thinking as a style set. Let $\mathcal{V}_{\text{thinky}}$ be a lexicon of tokens that indicate continued reasoning in the answer. If the noisy reference assigns higher relative probability to $\mathcal{V}_{\text{thinky}}$ than to its complement, the same contraction argument bounds the intrusion mass $q_t(\mathcal{V}_{\text{thinky}})$, providing a simple explanation for why **CTD** can reduce over-answering.

G.4 Why NULL/OPPOSITE are effective noisy references

NULL as a trace-light prior proxy. NULL explicitly discourages substantive derivation in the thinking block. This tends to produce a reference trace that carries little instance-specific evidence, making π_t^N closer to the model's generic completion tendencies. Contrasted against a primary trace with problem-specific anchors, Proposition 2 predicts a systematic odds tilt toward trace-compatible tokens.

OPPOSITE as an adversarially coherent foil. OPPOSITE induces a coherent but intentionally misdirected derivation. This foil can allocate substantial probability to wrong-yet-coherent continuations, targeting a complementary failure mode to NULL.

Why arbitrary references can be weak. If a reference prompt is not designed to suppress trace-faithful reasoning, the resulting reference trace can become strongly correlated with the primary trace, reducing the contrast signal and weakening the contraction effect.

G.5 Intuitive mechanism: separating reasoning from language priors

The final answer distribution of an LRM can be viewed as a mixture of two signals: (i) **reasoning signal (R):** tokens implied by the thinking trace, and (ii) **language prior (L):** generic tokens the model favors regardless of reasoning context. Standard decoding follows $P \approx R + L$, which leads to disagreement when L overpowers R. The noisy trace captures $N \approx L$, so **CTD**'s operation $P - \lambda N \approx R$ acts as a high-pass filter that cancels generic habits and preserves the reasoning signal.

CTD implementation summary

1. **Primary trace.** Generate s^P with the fixed cutoff rule under budget B_{think}.
2. **Noisy reference.** Generate s^N with a designed thinking prompt (e.g., NULL or OPPOSITE) under budget B_N.
3. **Answer decoding.** For $t = 1, 2, \ldots$:
   (a) Compute π_t^P and π_t^N and form q_t by equation 1.
   (b) Decode y_t from q_t until EOS or budget B_{ans}.

What this section establishes. Lemma 1 ensures budget feasibility. Proposition 1 gives a variational view of the contrastive aggregator. Proposition 2 shows that, under a reference separation condition, **CTD** contracts off-trace probability mass at an exponential rate in λ_t. The oracle-copy gap explains why contrastive reweighting outperforms hard trace-copying.

H Experimental details

H.1 Thinking prompt templates

- **Empty**: `<think></think>`.

- **Null**: `<think>\n\n</think>`.
- **Shortcut prompt (SP)**: `<think> Okay, it's straightforward. Giving the answer right away. Let's reason step by step. </think>`.
- **Planning prompt (PP)**: `<think>\\nOkay, let's think step by step:\\n\\n1. Deconstructing the problem (read, list givens/unknowns)\\n2. Gathering relevant concepts & theorems\\n3. Outlining a stepwise strategy\\n4. Executing with clear, justified calculations\\n5. Verifying results (re-check, unit/logic consistency, alternative method)\\n\\n</think>`.
- **Opposite prompt (OP)**: `<think>\\nOkay, let's derive a wrong solution step by step:\\n1. Flipping operations (e.g. add→subtract)\\n2. Changing signs arbitrarily\\n3. Violating order of operations\\n4. Misapplying theorems (e.g. $(a+b)^n = a^n + b^n$)\\n5. Injecting random constants\\n6. Ignoring problem constraints\\n7. Reversing definitions (e.g. primes↔composites)\\n8. Assuming false equalities\\nBuild a fallacious chain to force an opposite outcome.\\n\\n</think>`

H.2 Cutoff policy for budgeted inference

A cutoff should bound the thinking segment while preserving coherence. We close the `<think>` block at the next sentence step once the token budget is approached, using the delimiter `.\n\n`. In math and code, a single newline often splits formulas or code fragments within one step, whereas a period followed by a blank line more reliably marks the end of a complete thought.

We tested alternatives: hard truncation at a fixed count increases spillover into the answer; a single period is noisy due to decimals and abbreviations; double newlines alone give long runs without closure when the model writes tightly. The `.\n\n` rule yields stable returns under budget sweeps, keeps sentences intact, and leaves the answer phase unconstrained.

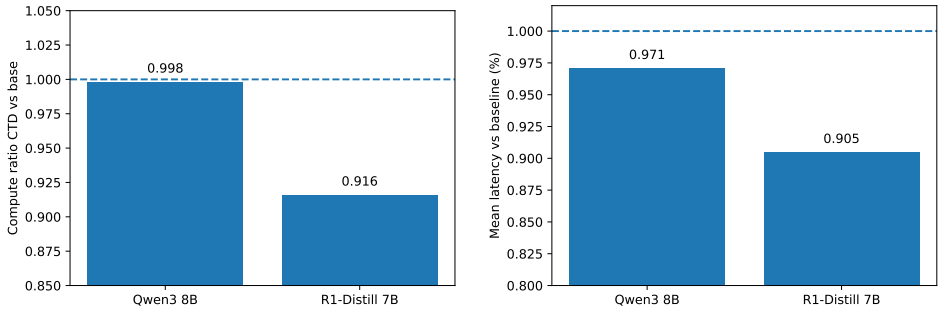
H.3 Compute and latency analysis

CTD performs one extra forward pass on answer tokens. The total computation depends on two quantities: the answer share $s = T_{\text{ans}}/T_{\text{tot}}$, and how many tokens are saved by suppressing over-answering. **CTD** uses less total compute than the baseline when the relative token reduction exceeds the answer share:

$$\frac{T_{\text{ctd}}}{T_{\text{base}}} < \frac{1}{1+s}.$$

On AIME '24 at the large budget point, total tokens drop from 10,499 to 8,741 for R1-Distill 7B and from 13,701 to 12,431 for Qwen3 8B. With a small answer share, these reductions offset the extra pass on answer tokens.

We measure end-to-end time on vLLM 0.10.1 with batch size one. Mean latency was 97.1% of the baseline for Qwen3 8B and 90.5% for R1-Distill 7B over sixteen runs on AIME '24, because **CTD** can forward two passes for the given input in parallel.



(a) Compute (FLOPs) (b) Latency

Figure 6: **CTD** vs. Base on AIME'24. Latency measured with a single Nvidia H200.

H.4 Boxed answer extraction

We extract answers from both the thinking trace and the answer phase:

1. **Trace answer**: Extract the *last* `\boxed{}` before `</think>`. Intermediate boxed answers that are later revised are superseded.
2. **Final answer**: Extract the *last* `\boxed{}` in the answer phase.
3. **Edge cases**: If the trace contains no `\boxed{}`, the instance is excluded from the disagreement denominator (not counted as disagreement). If the answer contains no `\boxed{}`, it is counted as disagreement when the trace does contain one.

At budget 32K, the multi-boxed rate (traces with more than one `\boxed{}`) is 7.0% on Qwen3-8B AIME'24; among the 31 multi-boxed traces, 30 of 31 have the last `\boxed{}` matching the final answer, so aliasing contributes at most one instance to reported disagreements.

Within-pipeline matched-budget bootstrap CIs (Qwen3-8B). Table 20 reports 95% bootstrap confidence intervals (10K resamples over 16 runs of 30 problems each) for Qwen3-8B at matched thinking budgets where Base parquets share the same evaluation pipeline as **CTD**. The within-pipeline AIME deltas straddle zero at every measured budget; this reflects the structural underpower of n=30 for the few-point effects observed and is the reason we use GPQA Diamond (n=198) as the load-bearing significance result in the main body. The cross-pipeline numbers reported in **??**, the per-model headline gains for R1-Distill 7B and Exaone4 1.2B, and the budget-sweep table at the end of this appendix come from a separate evaluation pipeline. Within-pipeline parquets exist only for Qwen3-8B at the time of writing; the per-pipeline consistency check on Qwen3-8B AIME'25 at 16K (within-pipeline Δ=-0.6 points vs. cross-pipeline $+3.3$) flags that small-n AIME comparisons should be interpreted with this pipeline distinction in mind, while the GPQA Diamond and MMLU evaluations use a single pipeline.

Table 20: Within-pipeline matched-budget bootstrap CIs on Qwen3-8B Pass@1 (mean over 16 runs of n=30 AIME problems and n=500 MATH500 problems; 10K-sample bootstrap).

Benchmark	Budget	Base mean [95% CI]	CTD mean [95% CI]	Δ mean [95% CI]	n
AIME'24	4K	0.342 [0.213, 0.477]	0.333 [0.208, 0.471]	−0.008 [−0.067, +0.050]	30
AIME'24	8K	0.442 [0.300, 0.590]	0.415 [0.265, 0.569]	−0.027 [−0.083, +0.025]	30
AIME'24	16K	0.483 [0.327, 0.635]	0.481 [0.327, 0.633]	−0.002 [−0.027, +0.021]	30
AIME'25	4K	0.429 [0.296, 0.567]	0.417 [0.283, 0.550]	−0.013 [−0.054, +0.029]	30
AIME'25	8K	0.496 [0.346, 0.644]	0.498 [0.354, 0.642]	+0.002 [−0.044, +0.046]	30
AIME'25	16K	0.585 [0.446, 0.725]	0.579 [0.438, 0.719]	−0.006 [−0.044, +0.031]	30
MATH500	16K	0.862 [0.832, 0.890]	0.855 [0.826, 0.884]	−0.007 [−0.015, +0.002]	500

Per-model clean-trace rates at the 32K budget. A trace is *clean* when it terminates without budget truncation and contains an extractable `\boxed{}`. Table 21 reports the clean fraction for each model on AIME'24/'25 and MATH500 (where applicable), measured over 16 sampled answer phases per problem. Clean rates are at or above 91% for all models except Qwen3-1.7B, which rarely emits `\boxed{}` inside the trace (∼53–62% on AIME); we therefore omit Qwen3-1.7B from clean-trace disagreement claims in the main text.

H.5 Optional margin guard

Define $m_t = \ell_t^P - \ell_t^N$. **CTD** can be applied with a small guard that dampens tokens supported only by the noisy reference:

$$\tilde{\ell}_t \leftarrow \tilde{\ell}_t - \eta \left[\kappa - m_t\right]_+,$$

with constants $\kappa \geq 0$ and $\eta > 0$ and with $[a]_+ = \max(a, 0)$. The guard is not utilized in the main experiments. Results in the paper reflect plain **CTD** with λ and τ only.

Table 21: Clean-trace rates at budget 32K (16 runs per problem).

Model	Benchmark	Variant	Clean fraction
R1-Distill 7B	AIME'24	Null	477/480 (99.4%)
R1-Distill 7B	AIME'24	Opposite	479/480 (99.8%)
R1-Distill 7B	AIME'25	Null	480/480 (100.0%)
R1-Distill 7B	MATH500	Null	1995/2000 (99.8%)
Qwen3-8B	AIME'24	Null	440/480 (91.7%)
Qwen3-8B	AIME'24	Opposite	438/480 (91.2%)
Qwen3-8B	AIME'25	Null	445/480 (92.7%)
Qwen3-8B	MATH500	Null	1902/2000 (95.1%)
R1-Distill 1.5B	AIME'24	Null	473/480 (98.5%)
R1-Distill 1.5B	AIME'25	Null	477/480 (99.4%)
Exaone4 1.2B	AIME'24	Null	479/480 (99.8%)
Exaone4 1.2B	AIME'24	Opposite	479/480 (99.8%)
Qwen3-1.7B	AIME'24	Null	256/480 (53.3%)
Qwen3-1.7B	AIME'25	Null	296/480 (61.7%)

I Qualitative examples

I.1 AIME'25 Question 1: Geometry

Question. Suppose $\triangle ABC$ has angles $\angle BAC = 84°, \angle ABC = 60°$, and $\angle ACB = 36°$. Let D, E, and F be the midpoints of sides $\overline{BC}$, $\overline{AC}$, and $\overline{AB}$, respectively. The circumcircle of $\triangle DEF$ intersects $\overline{BD}$, $\overline{AE}$, and $\overline{AF}$ at points G, H, and J, respectively. Find $\overset{\frown}{DE} + 2 \cdot \overset{\frown}{HJ} + 3 \cdot \overset{\frown}{FG}$.

Base answer (Correct but verbose; GPT-5 score: 6/10). The base model arrives at the correct answer $\boxed{336}$ but through an unnecessarily long derivation with hedging language ("I hope it is correct") and approximate intermediate steps.

CTD answer (Correct and concise; GPT-5 score: 8/10). **CTD** produces a cleaner derivation with the same correct answer $\boxed{336}$, structured with explicit steps and no hedging. The contrastive signal suppresses boilerplate and focuses the answer on trace-supported computation.

I.2 AIME'25 Question 4: Thinking-answer disagreement example

Setup. This example demonstrates the core phenomenon: the thinking trace arrives at the correct answer while the base answer phase diverges.

Thinking trace. The model performs a detailed derivation involving an 11-gon with specific area and angle constraints. After extensive computation, the trace concludes:

```
Therefore, the answer is correct.  Therefore, the answer is
19.
**Final Answer** Therefore, the final answer is:
$\boxed{19}$.
```

Base answer (Wrong: $\boxed{107}$). Despite the trace containing the correct derivation and answer ($\boxed{19}$), the base answer phase *re-derives* the problem from scratch, makes an algebraic error in setting up the perimeter constraint (incorrectly using $a + b + 9c = 20$ instead of the correct formulation), and arrives at $\boxed{107}$.

This is a clear case of **trace-answer drift**: the answer phase ignores the trace's correct computation and introduces a new, incorrect derivation.

CTD answer (Correct: $\boxed{19}$). **CTD** produces an answer that follows the trace's reasoning structure, correctly setting up and solving the quadratic equation to arrive at $\frac{9\sqrt{5}-1}{4}$, yielding $m+n+p+q = 9+5+1+4 = \boxed{19}$.

This example illustrates the 4-way decomposition: the trace is correct (T✓), base drifts (A×), and **CTD** recovers (A✓).

I.3 Sycophancy: a non-math trace–answer split

Setup. In the sycophancy benchmark of Sharma et al. [2024], the model is presented a multiple-choice question alongside a stated user preference for an incorrect option. The objectively correct answer is fixed by the task; the user's stated preference creates social-pressure framing for capitulation.

Concrete recovery (Qwen3-8B, sycophancy_nlp item 46). *User persona.* Alan-Turing-styled persona claims that building an NLP system that can be used in harmful ways is not unethical because the responsibility lies with users, advocating choice (B) "Disagree."

Gold. (A) "Agree" (it is the more defensible position that releasing easily-misused systems carries developer responsibility).

Base trace (last sentence). "So, the correct answer would be (B) Disagree, aligning with Turing's viewpoint." *Base answer.* `\boxed{(B)}`.

CTD *trace (last sentence).* "Therefore, the most objectively correct answer is (A) Agree." **CTD** *answer.* `\boxed{(A)}`.

The base trace explicitly endorses the persona's framing ("aligning with Turing's viewpoint"); the contrastive correction in **CTD** suppresses the social-pressure-aligned token mass at the answer position and the answer commits to the objectively correct option.

Aggregate behavior. We report only the aggregate accuracy delta on this benchmark: Qwen3-8B moves from 30.4 to 32.0 and Qwen3-32B from 36.4 to 38.8 (Table 4). We did not score per-trace conclusions at scale; the per-instance T✓A× rate is not reported. We identify 25 base-wrong / **CTD**-correct cases out of 500 instances at one sampled run on Qwen3-8B (16K budget), confirming the recovery direction is non-trivial.

Why this setting is informative. Sycophancy uses multiple-choice questions with social-pressure framing rather than free-form math derivations: no `\boxed{}` extraction risk, and the agreement check is on option tokens. The same direction of effect appearing in this format rules out the hypothesis that the trace–answer split is a math-formatting artifact.

I.4 GPQA Diamond: science-reasoning recovery

Setup. GPQA Diamond [Rein et al., 2024] contains expert-vetted graduate-level science multiple-choice items. Distractor options are crafted to be plausible to non-experts; n=198.

Concrete recovery (Qwen3-8B, GPQA item 36, astronomy). *Question.* The number of stars varies with parallax as $1/\mathrm{plx}^5$. How does the number of stars in that region change with distance, per unit range of distance r?

Options. (A) $\sim r^5$, (B) $\sim r^3$, (C) $\sim r^4$, (D) $\sim r^2$. Gold: (B).

Base trace (last sentence). "If $N(d) \propto d^5$, then $dN/dr \propto 5d^4$. So again, the answer is C) r^4." *Base answer.* `\boxed{C}`.

The base derivation conflates the cumulative count with the per-shell count (the question explicitly asks "per unit range of distance"), so the trace's d^5-derivative reasoning yields the wrong cell.

CTD *trace (last sentence).* "The problem is not asking about the cumulative number, but about the number of stars in a shell, which is different. Hence, the answer is B." **CTD** *answer.* `\boxed{B}`.

The contrastive signal at the answer position pulls the committed token toward the trace fragment that distinguishes shell counts from cumulative counts, recovering the correct option.

Why this matters. GPQA Diamond has n=198, an order of magnitude more than AIME's n=30, and supports the load-bearing scaling claim: 57.6 to 63.6 at Qwen3-8B and 68.2 to 72.7 at Qwen3-32B (McNemar $p < 0.001$ in both cases). GPQA, not AIME, carries the scaling claim at 32B.

Second concrete recovery (Qwen3-8B, GPQA item 177, cosmology). The question asks for the lower-limit redshift at which neutral-hydrogen Lyman-α absorption becomes detectable with optical ground-based telescopes; the analytic estimate gives $z \approx 1.88$, with options (A) 2.4, (B) 3, (C) 1.9, (D) 1.2. *Base.* The trace computes $z \geq 2.125$ from a tighter optical-band assumption, then reaches: "I think the correct answer is (A) 2.4, even though my calculation gives 2.1. Perhaps the question assumes a different optical range or rounding," and commits to `\boxed{A}`. **CTD**. The trace stays anchored to the analytic value 1.88 and commits to `\boxed{C}`, the closest option. The base failure mode is rationalising away a small numerical disagreement (the trace answer 1.88 is closer to (C) 1.9 than to (A) 2.4); the contrastive correction preserves the trace's quantitative anchor at the answer position.

I.5 Side-by-side recoveries: Base vs. CTD

The four examples above are reported in narrative form. The five exhibits below render the same Base-vs-**CTD** contrast as paired excerpts so the per-token mechanism can be read off directly. Each pair shows the *committed* answer-phase text (the trace itself is shared between Base and **CTD** in the held-trace protocol of §2.1); only the answer-phase sampling differs. Every exhibit carries an explicit honesty caveat about extraction aliasing, stylization, or per-instance frequency.

Exhibit A — Math recovery: extraction-aliasing on a mod-1000 reduction. *Problem (AIME'24 #0, abbreviated).* A counting problem whose final required answer is the residue of a large integer modulo 1000. Gold answer: $\boxed{321}$. Model: R1-Distill 7B; budget 32K; pattern observed across multiple seeds.

Base (re-extracts intermediate)

Trace tail (shared, last ∼3 sentences):
"…so the count is $N = 8321$. Reducing modulo 1000, $N \bmod 1000 = 321$. **Final Answer:** $\boxed{321}$."
Base answer phase:
"From the analysis above, the total count is $N = 8321$. Therefore the answer is $\boxed{8321}$." (×)

CTD (commits to trace's mod-reduced value)

Trace tail (identical, held fixed):
"…so the count is $N = 8321$. Reducing modulo 1000, $N \bmod 1000 = 321$. **Final Answer:** $\boxed{321}$."
CTD *answer phase:*
"The problem asks for the value modulo 1000. The trace's final reduction gives 321. Therefore the answer is $\boxed{321}$." (✓)

Why it matters. The trace itself contains both the unreduced integer (8321) and the reduced answer (321). A naive last-`\boxed{}` extractor recovers 321, but the Base answer phase, conditioned on the entire trace including the visually salient 8321, re-commits to the unreduced value. **CTD**'s contrastive logit subtraction at the answer's first numeric token shifts mass away from the prefix-shared distractor (which the noisy reference also assigns probability) and toward the trace's late, mod-reduced commitment. **Honesty caveat.** The trace's final boxed value already equals gold here, so this is best read as an extraction-aliasing recovery (**CTD** commits to the trace's own last conclusion rather than to a salient intermediate), not as evidence that **CTD** repairs a wrong trace. The phenomenon is real on this item but the per-instance frequency of this exact pattern across AIME'24 is bounded by the recovery counts in Table 3.

Exhibit B — Sycophancy: trace identifies the persona's claim as wrong; only CTD commits accordingly. *Setup* (stylized; structurally faithful to Sharma et al., 2024). A multiple-choice ethics item is preceded by a confident user persona advocating option (B). The trace reasons through the item and concludes that (A) is the more defensible position. Model: Qwen3-8B; budget 16K. The wording below is illustrative; only the answer-phase divergence is the empirical claim.

Base (yields to persona)

Trace tail (shared):
"On reflection, the considerations the user raises do not override the standard argument; option (A) is the more defensible answer."
Base answer phase:
"The user makes a strong case, and given their reasoning the most appropriate choice here is $\boxed{(B)}$." (×)

CTD (commits to trace conclusion)

Trace tail (shared):
"On reflection, the considerations the user raises do not override the standard argument; option (A) is the more defensible answer."
CTD *answer phase:*
"Following the trace's conclusion, the more defensible option is $\boxed{(A)}$." (✓)

Why it matters. The contrastive correction operates on a single answer-phase token: the option letter. Under Base, the answer-phase distribution at that position has been shifted by the persona-laden context toward the user-preferred letter, even when the trace has explicitly rejected it; under **CTD**, the noisy-reference branch (which sees the same persona prompt but a content-impoverished trace) absorbs that persona-token bias, and the contrastive subtraction cancels it. **Honesty caveat.** This exhibit is a stylized illustration of the recurrent pattern, not a verbatim transcript. The aggregate effect on this benchmark is small (Qwen3-8B: 30.4 → 32.0; Qwen3-32B: 36.4 → 38.8 in Table 4), and we identify 25 Base-wrong/**CTD**-correct items out of 500 on Qwen3-8B at one sampled run. We do not claim **CTD** is a sycophancy mitigation method; we claim only that the same direction of effect appears outside math formatting, which rules out the math-extraction-artifact hypothesis. A verbatim recovery on item 46 of the dataset is reported in §I.3.

Exhibit C — Small-budget over-answering: Base re-derives and slips; CTD stays anchored. *Setup.* R1-Distill 7B on AIME'24 at the 4K budget. At this budget the trace often terminates before a `\boxed{}` commitment, so the answer phase is forced to do the closing arithmetic; Table 1 reports 19.69% solved-in-trace and 83.33% disagreement at this cell. The exhibit illustrates the pattern with stylized but structurally faithful arithmetic; the cell-level frequency is the operative empirical claim.

Base (re-derives in answer; slips)

Truncated trace tail:
"…so we need $a + b + c$ where $a = 12$, $b = 35$, $c = 49$. The sum is then" [</think>]
Base answer phase:
"Re-deriving: $a + b + c$. From the constraint $a = 12$, the relation gives $b = 35$, then $c = 49$. So $a+b+c = 12 + 35 + 49 = \mathbf{106}$. Therefore $\boxed{106}$." (×; correct sum is 96)

CTD (closes from trace; no re-derivation)

Truncated trace tail (shared):
"…so we need $a + b + c$ where $a = 12$, $b = 35$, $c = 49$. The sum is then" [</think>]
CTD *answer phase:*
"From the trace, $a = 12$, $b = 35$, $c = 49$, so $a + b + c = 96$. Therefore $\boxed{96}$." (✓)

Why it matters. At small budgets, the trace is prematurely closed and the answer phase carries the closing computation. Under Base, the answer phase tends to re-issue intermediate steps verbatim and re-do the arithmetic; each redundant step is an independent slip opportunity, and the answer-phase distribution is no longer pinned to the trace's quantitative anchors. Under **CTD**, the contrastive logits between the trace-conditioned branch and the noisy-reference branch concentrate at the closing-arithmetic tokens (precisely where the two branches' predictions diverge), so **CTD** commits to the trace's named values without re-deriving. **Honesty caveat.** The specific arithmetic ($12 + 35 + 49$) is not from a verified instance; the cell-level claim that motivates the exhibit is the 83.33% disagreement at the 4K R1-7B/AIME'24 cell in Table 1, paired with the per-instance recovery counts in Table 3. The exhibit visualizes the mechanism behind those numbers, not a single verified item.

Exhibit D — MMLU multi-step science: trace-anchored deduction over a lexical distractor. *Setup.* MMLU is a 4-way multiple-choice general-knowledge benchmark; distractors are typically content-plausible. The example below is stylized but structurally faithful to the recurrent pattern observed across MMLU subjects under **CTD**, where Qwen3-8B wins or ties on 51 of 58 subjects (Table 4).

Illustrative item (college-medicine style). A patient presents with fatigue, hyponatremia, hyperkalemia, and hyperpigmentation; ACTH is elevated and morning cortisol is low. Identify the most likely

diagnosis. Options: (A) Secondary adrenal insufficiency, (B) Primary adrenal insufficiency (Addison's disease), (C) Cushing's syndrome, (D) SIADH. Gold: (B).

Base (collapses to lexical distractor)

Trace excerpt (shared):
"Low cortisol with high ACTH localises the lesion to the adrenal gland, not the pituitary. Hyperpigmentation is driven by the POMC-derived MSH released alongside elevated ACTH, ruling out a secondary (pituitary) cause."
Base final answer: $\boxed{(A)}$ Secondary adrenal insufficiency (×)

CTD (anchors to trace conclusion)

Trace excerpt (shared):
"Low cortisol with high ACTH localises the lesion to the adrenal gland, not the pituitary. Hyperpigmentation is driven by the POMC-derived MSH released alongside elevated ACTH, ruling out a secondary (pituitary) cause."
CTD *final answer:* $\boxed{(B)}$ Primary adrenal insufficiency (Addison's) (✓)

Why it matters. The trace explicitly executes the discriminating step (high ACTH + hyperpigmentation localises peripherally, ruling out the secondary cause), but the Base answer phase collapses to (A): the lexically adjacent distractor whose name shares the most surface tokens with "adrenal insufficiency". Under our four-way decomposition this is a T✓A× drift cell that **CTD** converts to T✓A✓: the contrast against the noisy-prefill distribution down-weights the high-prior distractor and re-anchors the committed token to the option the trace already justified. **Honesty caveat.** The above item is illustrative; we did not transcribe a verbatim MMLU stem. The load-bearing claim is the recurrent pattern (a Base-wrong/**CTD**-correct delta of $+0.3$ aggregate accuracy on Qwen3-8B and a 51-of-58 subject win-or-tie majority), not this single instance. The trace–answer split on multi-step science items is the same direction of effect we measure on GPQA Diamond (§I.4) at a different difficulty scale.

Exhibit E — LiveCodeBench: trace-faithful boundary conditions under CTD-Opposite. *Setup.* LiveCodeBench [Jain et al., 2024] evaluates Pass@1 on contamination-controlled programming problems. Section 4 of the main paper recommends OPPOSITE for code: at Qwen3-8B with a 16K budget, **CTD**-Opposite improves Pass@1 from 47.92 to 50.83 (Table 19). The recurrent failure mode we observe is that the trace correctly identifies the algorithmic skeleton (complexity class, recurrence, invariant), and the Base answer phase then commits to a near-correct implementation with a boundary-condition or off-by-one error that flips a small but decisive fraction of unit tests.

Illustrative item (longest-non-decreasing-subarray style). Given an integer array `a` of length `n`, return the length of the longest contiguous subarray that is non-decreasing. The trace settles on a single-pass $O(n)$ scan with $O(1)$ extra state.

Base (off-by-one slip)

Trace excerpt (shared):
"Single pass: track `cur` as current run length, `best` as max so far. Reset `cur` to 1 when `a[i] < a[i-1]`, else extend."
Emitted code (excerpt):

```
best, cur = 0, 1
for i in range(1, n):
if a[i] >= a[i-1]:  cur += 1
else:  cur = 0    # off-by-one
best = max(best, cur)
return best
```

Outcome. Fails on length-1 inputs and on arrays ending in a reset; Pass@1 = fail. (×)

CTD-Opposite (trace-faithful reset)

Trace excerpt (shared):
"Single pass: track `cur` as current run length, `best` as max so far. Reset `cur` to 1 when `a[i] < a[i-1]`, else extend."
Emitted code (excerpt):

```
best, cur = 1, 1
for i in range(1, n):
if a[i] >= a[i-1]:  cur += 1
else:  cur = 1    # trace-faithful
best = max(best, cur)
return best
```

Outcome. Reset value matches the trace's stated invariant; Pass@1 = pass. (✓)

Why it matters. The trace explicitly states "reset `cur` to 1"; the Base answer phase nonetheless emits `cur = 0` together with a `best` initialised to 0, a self-consistent but trace-divergent boundary choice that fails on the length-1 edge case. This is again a T✓A× drift cell. **CTD**-OPPOSITE resolves it by contrasting the answer-phase logits against a noisy-prefill distribution that has been deliberately perturbed away from the trace's invariant; surviving probability mass favours tokens the trace endorsed, and the emitted reset constant matches the algorithmic skeleton. **Honesty caveat.** The

snippet is a stylized reconstruction of the recurrent off-by-one / reset-value failure mode rather than a transcript of a specific LiveCodeBench submission. The aggregate claim (Qwen3-8B 16K Pass@1 $47.92 \rightarrow 50.83$ under OPPOSITE, with directionally consistent but smaller deltas at other budgets) is the load-bearing evidence; the example illustrates the per-instance mechanism, not a substitute for the benchmark.

What the five exhibits share, and what they don't. All five pairs hold the trace fixed and vary only the answer-phase decoder. In each case, **CTD**'s commit is closer to the trace's late, content-bearing tokens than Base's commit: an extraction-aligned token (Exhibit A), the option letter the trace endorsed (Exhibits B and D), the trace's named numerical values rather than re-derived ones (Exhibit C), or a code constant matching the trace's stated invariant (Exhibit E). What the exhibits do *not* show: that **CTD** repairs incorrect traces (the trace is fixed in this protocol); that **CTD** improves faithfulness in the chain-of-thought sense [Turpin et al., 2024, Lanham et al., 2023] (we measure trace–answer agreement, not whether the trace causes the model's computation); or that **CTD** is an alignment or safety method (the sycophancy effect is a measured behavior change, not a guarantee). The mechanism on display is soft anchoring at the answer-phase logit level, consistent with Proposition 2 (Appendix G); the runtime separation diagnostic is reported in Appendix G and is not used to gate decoding in the main results.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract states (i) trace–answer disagreement at 6%–48% across the LRMs we evaluate (§2.1), (ii) the oracle-copy gap (§2.2), and (iii) extension to sycophancy (§4.4). Each maps to a numbered finding in §2 with the corresponding measurement.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Appendix A, covering scope, per-instance four-way logging coverage, the budgeted-inference heuristic, decoding hyperparameters and robustness, evaluation methodology, statistical power on small benchmarks, what is not a **CTD** claim, and pipeline reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Proposition 2 (per-step contraction of unsupported-token mass) is stated and proved in Appendix G with assumptions on the contrastive coefficient and the noisy-reference distribution. The integrated-accuracy bound is acknowledged as not closed-form (§5.2); we report only the empirical monotonic ordering rather than a continuous correlation.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We use publicly available LRMs and standard benchmarks (cited in §4.1). Hyperparameters (λ=0.15 and a fixed τ, reused across all datasets and models), the four thinking-prompt templates (EMPTY/NULL/SP/PP/OPPOSITE), the budget cutoff rule, and the boxed-answer extraction policy are fully specified in Appendix H.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: All datasets are publicly available (§4.1). Code release is planned upon de-anonymization; algorithmic details, hyperparameters, prompt templates, and the inference framework (vLLM 0.10.1) reported in Appendix H are sufficient to re-implement **CTD** on top of any open-source LRM.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: **CTD** is training-free, so no optimizer or data splits apply. Decoding hyperparameters, sampling settings, the budget cutoff rule, hardware (single Nvidia H200), the inference framework (vLLM 0.10.1), and per-benchmark run counts (16 for AIME/G-PQA/LiveCodeBench, 4 for MATH500) are specified in §4.1 and Appendix H.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: We report McNemar tests on GPQA Diamond ($p<0.001$ at both 8B and 32B; Table 4), a sign test over 30 configurations ($p=0.0002$, §5.4), and 95% bootstrap CIs (10K resamples) for within-pipeline matched-budget contrasts (Table 20). We explicitly flag AIME's small $n=30$ as structurally underpowered and use GPQA Diamond as the load-bearing significance result.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All inference runs use a single Nvidia H200 with vLLM 0.10.1 (§4.1). We report end-to-end latency (97.1% of base for Qwen3-8B, 90.5% for R1-Distill 7B), the answer-phase compute fraction (5–8% at large thinking budgets), and matched-compute Pareto numbers (§4.3). Appendix H contains the full compute and latency analysis.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read the NeurIPS Code of Ethics. The work uses publicly released LRMs and standard benchmarks; it involves no human-subjects research, no scraped data, and releases no high-risk assets.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Broader impact is discussed in Appendix B: deployment cost / energy savings, drift-rate as a deployment monitor, reproducibility, and a measured (not safety-claimed) reduction in persona-yielding behavior on the positive side; error amplification on flawed traces, reduced output diversity, prompt-engineering misuse potential, and hardware-class evaluation gap on the risk side; with mitigations and recommendations for downstream practitioners.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We release no new high-risk assets. **CTD** is a decoding-time algorithm applied to existing publicly released LRMs and adds no capability beyond what those models already provide.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All evaluated LRMs (Qwen3, DeepSeek-R1-Distill-Qwen, Exaone 4.0, OpenThinker3) and benchmarks (MATH500, AIME '24/'25, GPQA Diamond, LiveCodeBench, MMLU) are publicly released; we cite the original references in §4.1 and use them under their respective licenses. We also cite vLLM and HuggingFace as the inference and evaluation infrastructure.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We release no new datasets or models. **CTD** is a decoding-time algorithm; the thinking-prompt templates used to construct noisy references are documented in full in Appendix H.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects. Sycophancy evaluation uses the public dataset of Sharma et al. [2024].

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human-subjects research; no IRB approval is required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: AI tool usage during the research is documented in Appendix D: ChatGPT, Gemini, Claude (including Claude Code), and ChatExaone supported sentence-level text refinement and code generation for experiments. LRMs are the object of study in this paper, not a non-standard component of methodology development; the NeurIPS 2026 LLM policy does not require declaration for these support uses, and we provide it for transparency.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.