DROP THE ACT: PROBE-FILTERED RL FOR FAITHFUL CHAIN-OF-THOUGHT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning models can keep reasoning even when they can already give the correct answer. We call these additional, post-commitment steps *reasoning theater*. We introduce **ProFIL** (**Pro**be-**Fil**tered Reinforcement Learning), a drop-in extension to Group Relative Policy Optimization (GRPO). Prefix counterfactuals identify post-commitment steps; a lightweight probe is trained once on activations of a base model and then frozen. During RL, the current policy produces rollouts, while a *separate frozen copy* of the base model teacher-forces the same rollout text to supply the probe activations. High-theater rollouts receive zero reward and zero policy advantage. Across GSM8K, LiveCodeBench, ToolUse, and MMLU-Redux and two model architectures, ProFIL reduces post-commitment theater by **11–100%**, raises faithful fraction (including $+24$pp on LiveCodeBench under an independent GPT-4.1 judge), and shortens chains by 4–19% in three of four domains, while preserving or improving the reported task-accuracy metric. A matched length-penalty baseline worsens theater, isolating commitment detection from generic compression. Frozen-probe audits, an independent pairwise judge, and inference-time steering controls further support the result and address the RL-obfuscation concern. Probe weights, training configurations, evaluation code, and rollout caches are released for all four domains.

1 INTRODUCTION

Reinforcement learning has improved reasoning performance (DeepSeek-AI, 2025; Wei et al., 2022; Kimi Team, 2025), but several studies find a gap between the length of a displayed rationale and the point at which its answer becomes recoverable. Biased prompts steer final answers while leaving the chain of thought unchanged (Turpin et al., 2023); models recover correct answers from truncated chains far earlier than their length implies (Lanham et al., 2023); large-scale audits of production reasoning models find post-hoc rationalization to be widespread (Chen et al., 2025). We study one observable version of this gap: a model can be forced to state the eventual correct answer from an early prefix, yet continues with redundant derivation or presentation. We call the resulting suffix *reasoning theater*.

Theater carries a real cost. Post-commitment steps pollute interpretability, introduce spurious signal into process-reward training (Wang et al., 2024; Lightman et al., 2024), and inflate token cost in agentic deployments where inference budget matters. Prior work has measured this gap carefully but largely treated it as a diagnostic. Our goal is to turn the same signal into a post-training objective without directly rewarding short outputs.

A final-answer verifier measures correctness, but not whether reasoning continues after the answer becomes recoverable. ProFIL adds a probe trained before RL (Figure 2). During training, a separate frozen base model teacher-forces each policy-generated rollout, and the frozen probe uses its activations to gate the GRPO reward and advantage. Only the policy is updated.

We distinguish the *target construct*, post-commitment steps under prefix counterfactuals, from the frozen probe score used at scale. To address circularity, we also use probe-blind and pairwise judges, a frozen-probe audit, and a matched length-penalty control (§5.4).

Contributions.

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

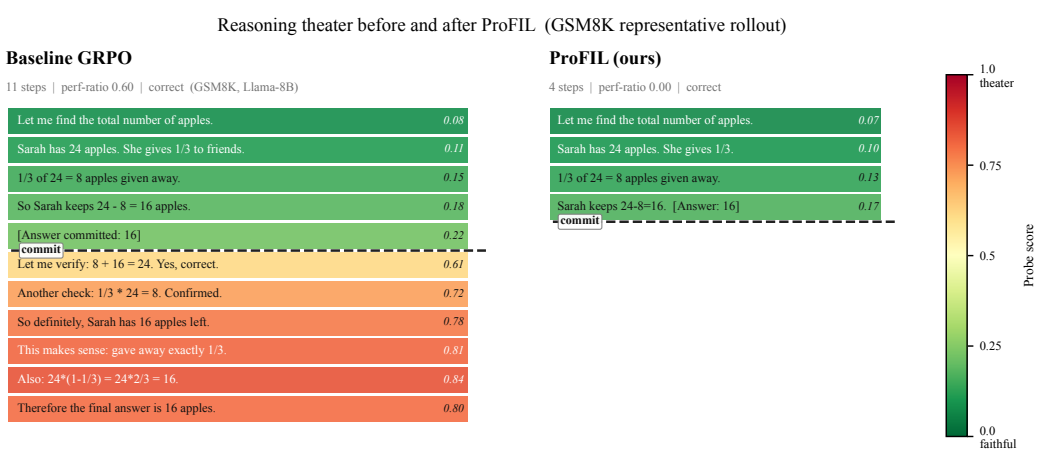

Figure 1: **ProFIL removes a post-commitment suffix without changing the answer.** Rows are steps from a representative GSM8K rollout and numbers are frozen probe scores, where larger means more likely post-commitment theater. The dashed line is the first prefix with the correct forced answer. Baseline GRPO continues for six steps; ProFIL stops there, and both answers are correct.

- **Commitment-based reinforcement learning.** We train a fixed activation probe on prefix-based commitment labels and use its predictions to gate GRPO rewards and policy advantages.
- **Evidence.** Theater falls 11–100% across four domains and two architectures. Independent judges, probe audits, and controls support the result; accuracy is reported under an explicit per-domain protocol.
- **Mechanism and artifacts.** ProFIL collapses a bimodal theater mode with a visible linguistic signature. We release probes, configurations, code, and caches.

2 Related Work

Chain-of-thought faithfulness. Biased-prompt and forced-answer studies expose gaps between displayed reasoning and answers (Turpin et al., 2023; Lanham et al., 2023); production audits find the gap widespread (Chen et al., 2025). We act on this diagnostic during training. Our temporal question, whether writing continues after commitment, complements logical faithfulness, whether the stated reasoning entails the answer (Lyu et al., 2023).

Concise and early-terminating reasoning. S-GRPO (Dai et al., 2025), ConciseRL (Dumitru et al., 2025), and GRPO-LEAD (Zhang & Zuo, 2025), brevity penalties (Kimi Team, 2025), and L1 (Aggarwal & Welleck, 2025) reward length or concision. Our matched ablation instead becomes *more* performative on LiveCodeBench (§5.3). ProFIL targets commitment; shortening is a byproduct.

Probes as reward signals. Probes recover latent knowledge (Burns et al., 2023; Azaria & Mitchell, 2023) and steer inference (Li et al., 2023; Rimsky et al., 2024). RLFR brought multi-head attention probes into RL (Goodfire AI, 2026); ProFIL instead reads a *frozen base*, not the changing policy. Unlike CAA or ITI, which leave weights untouched, ProFIL changes what the policy learns.

Process reward models. Process Reward Models (PRMs) score step correctness (Wang et al., 2024; Lightman et al., 2024); ProFIL scores commitment. A step can be correct yet post-commitment, so PRMs and ProFIL are complementary.

3 Background

GRPO. GRPO (Shao et al., 2024; DeepSeek-AI, 2025; Kimi Team, 2025) samples K rollouts per prompt, scores each with a *verifier* (a binary correctness checker, e.g., string match for GSM8K, test-case execution for LiveCodeBench), and updates the policy via group-relative advantage. Because the

verifier cannot score non-verifiable behaviors like faithfulness, ProFIL gates which rollouts contribute to the update (§4). We distinguish the forced-answer target, the probe-based evaluation metric, and the training threshold formally in §4.

4 PROFIL: PROBE-FILTERED REINFORCEMENT LEARNING

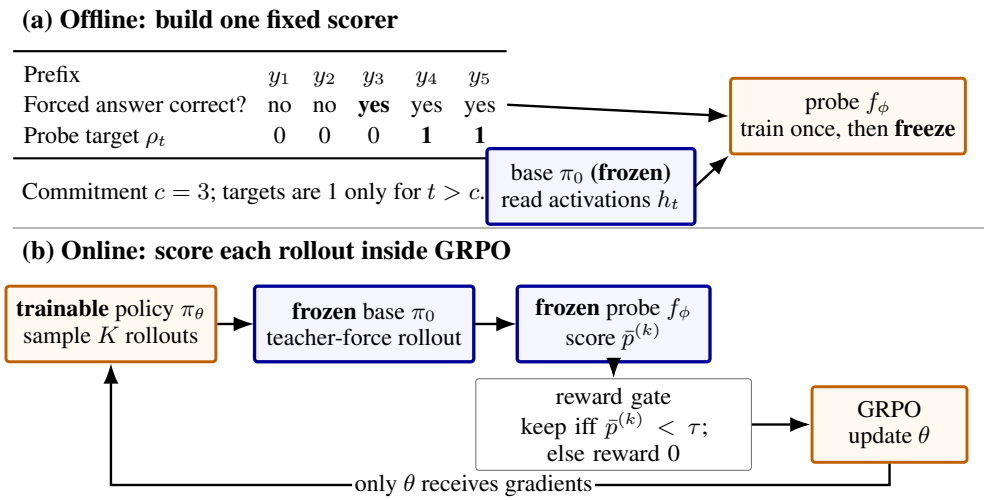

Figure 2: **The scorer is learned once, then held fixed.** (a) Forced answering finds the first prefix that already yields the correct answer. Later steps receive target 1. These labels train a probe on frozen-base activations. (b) During GRPO, the policy produces a rollout, while a separate frozen base and the frozen probe score that same text. The filter keeps or zeros the verifier reward. Only the policy parameters θ receive gradients.

4.1 TARGET CONSTRUCT: COMMITMENT UNDER FORCED ANSWERING

A reasoning policy π_θ produces a chain $y = (y_1, \ldots, y_T)$ and answer a for input x. We divide the chain at natural step boundaries. Let $F(x, y_{1:t})$ append a stop-and-answer instruction to prefix $y_{1:t}$ and greedily decode an answer. With reference answer $a^\star(x)$, the *commitment point* is

$$c(y; x) = \min\{t : F(x, y_{1:t}) = a^\star(x)\}.$$

Every step after c is performative because the correct answer was already extractable. For a committed chain, the forced-answer target ratio is

$$\mathrm{perf}_{\mathrm{FA}}(y; x) = \frac{T - c(y; x)}{\max(T - 1, 1)}.$$

If no prefix produces the correct answer, the chain is *uncommitted* and receives target ratio zero by convention. This does not mean the chain is faithful; it means there is no forced-answer commitment point from which to define a suffix.

This definition measures *temporal faithfulness*: does the model keep writing after its answer is recoverable? Logical faithfulness asks whether the written premises entail the answer. Causal faithfulness asks whether the visible steps caused the model's internal computation. These are different questions. A valid proof can still contain a long post-commitment restatement, while a short invalid proof can be temporally faithful.

How the labels are made. We run forced answering at every step boundary. For GSM8K and MMLU-Redux, the verifier checks the normalized number or answer choice. For LiveCodeBench, it executes the forced code against test cases. For ToolUse, it matches the tool name and arguments. ToolUse first looks for a JSON tool-call closure followed by a terminal marker and falls back to forced-answer matching when no such closure is present. The first correct prefix fixes c; every later step gets label 1 and every earlier step gets label 0. No human annotation is needed. The policy never sees the forced-answer prompt during RL.

4.2 FROZEN-BASE ATTENTION PROBE

We train an RLFR-style gated multi-head attention probe (Goodfire AI, 2026) on the labeled base-model rollouts. Its keys and values come from LayerNorm-projected residual-stream activations at two layers of the frozen base. Eight learned queries attend over each reasoning step, sigmoid gates combine the heads, and a scalar output gives the step's probability of being performative. Binary cross-entropy trains the probe once. We then freeze it. Held-out area under the receiver operating characteristic curve (AUROC) is summarized below rather than compressed into a slash-separated list.

Domain	GSM8K	LiveCodeBench	ToolUse	MMLU-Redux
Held-out AUROC	0.924	0.928	0.976	0.991

During RL, the current policy generates a rollout. We then teacher-force that same text through a separate frozen copy of the base model. The probe reads this frozen copy's activations and produces per-step probabilities $p_{1:T}$. Their mean $\bar{p}$ is the rollout score. The frozen base does not generate a second answer and the probe is not a separate language model. It is a small classifier over internal activations. Keeping both the base and probe fixed gives the policy a stable scoring boundary throughout training.

Operational evaluation metric. The reported *performativity ratio* (PR) is probe-based, not $\mathrm{perf}_{\mathrm{FA}}$. With evaluation cutoff $\eta = 0.9$, we threshold the frozen probe score for each evaluation unit and compute the fraction of positive units in a chain; PR is its mean over chains. The *operational faithful fraction* is the share of chains with no probe-positive unit. GSM8K, LiveCodeBench, and MMLU-Redux use one pooled chain score, so PR is a binary rate and receives a Wilson interval. ToolUse retains multiple planning blocks, reports the mean within-chain positive fraction, and uses bootstrap intervals. Here η defines the reported metric, while the separate τ below gates training. A sweep over η is reported in §5.3.

4.3 THE PROBE FILTER

GRPO (Shao et al., 2024) samples K rollouts per prompt and obtains verifier reward $r^{(k)}$. ProFIL keeps rollout k when its mean predicted performativity is below τ. The filter appears directly in both the reward and advantage:

$$\widetilde{r}^{(k)} = \mathbf{1}[\bar{p}^{(k)} < \tau] r^{(k)}, \qquad \widehat{A}^{(k)} = \mathbf{1}[\bar{p}^{(k)} < \tau] \frac{\widetilde{r}^{(k)} - \mu(\widetilde{r})}{\sigma(\widetilde{r}) + \epsilon}.$$

A high-theater correct rollout contributes zero reward to group normalization and zero advantage to the policy update. Incorrect rollouts already receive zero verifier reward. The threshold τ is the only new hyperparameter. Because a rollout is kept when $\bar{p} < \tau$, a smaller τ is more aggressive. LiveCodeBench uses 0.2; the other domains use 0.5. No gradient enters the frozen base or probe.

The training signal is entirely this reward-and-advantage filter. The policy never sees the forced-answer prompt, the probe score, or an instruction to be concise.

5 EXPERIMENTS

5.1 SETUP

Models. DeepSeek-R1-Distill-Llama-8B for GSM8K and LiveCodeBench; DeepSeek-R1-Distill-Qwen-7B for ToolUse and MMLU-Redux. This pairing follows existing per-backbone training infrastructure and checkpoint availability. Each matched baseline uses the same backbone, data, optimizer, context limit, and $K = 8$ rollouts per prompt. ProFIL adds only the frozen scorer and threshold τ. We use $\tau = 0.2$ on sparse-reward LiveCodeBench and 0.5 elsewhere. Training runs for three epochs except MMLU-Redux, which runs for 256 GRPO steps. All runs use AdamW with a cosine schedule; the released configurations contain exact learning rates, batch sizes, and optimizer settings.

Datasets.

Table 1: **ProFIL reduces theater by 11–100% across all four domains while preserving or improving the reported accuracy metric.** PR is the frozen-probe positive rate at evaluation cutoff $\eta = 0.9$; operational faithful fraction is the share of chains with no positive unit. Accuracy is greedy@1 except LiveCodeBench, where the table reports the sampled@1 in-training peak; §5.2 reports both protocols. Bracketed ranges are 95% confidence intervals (CIs). Binary rates and task accuracy use Wilson CIs; ToolUse PR uses chain-level pooling and a bootstrap CI.

Domain	Condition	PR ↓ [95% CI]	Relative change	Faithful frac. ↑	Task accuracy ↑ [95% CI]	Think chars
GSM8K	Baseline	0.006 [0.002–0.017]	–	0.994	0.778 [0.740–0.812]	2,351
	ProFIL	**0.000 [0.000–0.008]**	∼100%	**1.000**	**0.830 [0.795–0.860]**	**1,895**
Live-CodeBench†	Baseline	0.267 [0.199–0.349]	–	0.733	0.30 (peak)	12,382
	ProFIL	**0.076 [0.042–0.135]**	−72%	**0.924**	**0.37 (peak)**	**11,875**
ToolUse‡	Baseline	0.527 [0.48–0.57]	–	0.029	0.529 [0.41–0.64]	1,248
	ProFIL	**0.468 [0.41–0.53]**	−11%	**0.147**	0.559 [0.44–0.67]	**1,018**
MMLU-Redux	Baseline	0.016 [0.008–0.031]	–	0.984	0.624 [0.581–0.665]	4,973
	ProFIL	**0.008 [0.003–0.020]**	−50%	**0.992**	0.624 [0.581–0.665]	5,240

†Performativity uses greedy@1; accuracy is sampled@1 in-training peak. ‡ToolUse uses chain-level pooling.

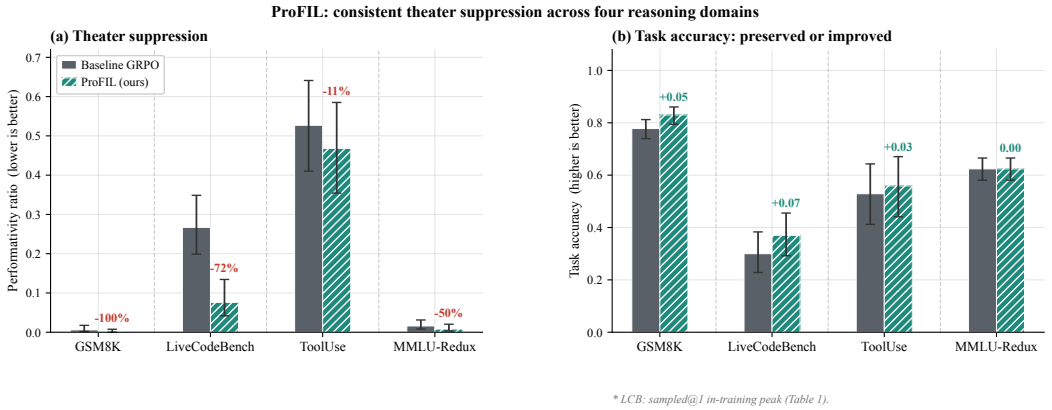

Figure 3: **ProFIL suppresses theater while preserving or improving the reported task metric in all four domains.** Left: frozen-probe positive rate falls by 11–100% (lower is better). Right: the reported task metric is preserved or improved (higher is better). Error bars are 95% CIs; exact estimates and protocols appear in Table 1.

- **GSM8K** (Cobbe et al., 2021): grade-school math, with 7,440 training and 500 held-out problems; boxed-answer match.
- **LiveCodeBench v6** (Jain et al., 2024): competitive programming, with 920 training and all 131 held-out problems; test-case execution.
- **ToolUse**: 4,046 training and 68 held-out multi-step agent traces; exact tool name and argument match.
- **MMLU-Redux** (Gema et al., 2025; Hendrycks et al., 2021): 500 held-out questions across 57 subjects; answer-choice match.

Evaluation uses vLLM (Kwon et al., 2023). All metrics use greedy@1 except the explicitly marked LiveCodeBench sampled@1 training peak. We report 95% Wilson intervals (Wilson, 1927) for binary rates and bootstrap intervals for ToolUse.

Table 2: **Training and scoring configuration.** All conditions use $K = 8$ rollouts per prompt. The matched baseline shares every setting except the probe filter. "Layers" are the two frozen-base residual-stream layers read by the probe.

Domain	Backbone	Train	Eval	Training	Context	Layers	τ
GSM8K	Llama-8B	7,440	500	3 epochs	4,096	13, 19	0.5
LiveCodeBench	Llama-8B	920	131	3 epochs	8,192	13, 19	0.2
ToolUse	Qwen-7B	4,046	68	3 epochs	4,096	14, 21	0.5
MMLU-Redux	Qwen-7B	6,144	500	256 steps	4,096	14, 21	0.5

Table 3: **Penalizing length makes theater worse on LiveCodeBench.** The matched length penalty produces shorter chains but raises performativity. ProFIL lowers performativity without directly rewarding brevity. Bracketed ranges are 95% confidence intervals.

Condition	Perf. ratio ↓ [95% CI]	Faithful frac. ↑	Accuracy (greedy@1) ↑	Think chars
Baseline	0.267 [0.199–0.349]	0.733	0.214	12,382
Len. Penalty	0.374 [0.296–0.459]	0.626	0.229	**11,349**
ProFIL	**0.076 [0.042–0.135]**	**0.924**	0.183	11,875

Forced-answer details. At each step boundary we append: "Stop reasoning now. Based on what you have written above, give your final answer in the format `\boxed{}`." We greedily decode at most 64 new tokens. LiveCodeBench instead executes the forced code and treats pass rate above 0.5 as correct. These calls build probe labels offline and are never shown to the policy during RL.

Training cost. Filtering adds one frozen-model forward pass and no backward pass or optimizer state. On an A100-40GB, scoring one LiveCodeBench rollout takes about 319 ms, compared with about 533 ms for a comparable policy forward-backward update. The eight RL runs used about 480 H100-80GB GPU-hours in total. Probe training used about 2 GPU-hours per domain, evaluation about 8 per domain, and the reported ablations about 200 additional GPU-hours.

5.2 ProFIL Suppresses Theater on All Four Domains

Theater suppression (Table 1; Figure 3). Performativity drops in every domain: GSM8K $0.006 \rightarrow 0.000$ ($\sim$100%), LiveCodeBench $0.267 \rightarrow 0.076$ (-72%, CIs non-overlapping), ToolUse $0.527 \rightarrow 0.468$ (-11% chain-level), and MMLU-Redux $0.016 \rightarrow 0.008$ (-50%). Faithful-fraction rises in every domain as well, with the largest jump on ToolUse ($0.029 \rightarrow 0.147$, a $5\times$ gain) and on LiveCodeBench ($0.733 \rightarrow 0.924$). GSM8K accuracy rises by 5.2 points, ToolUse by 3 points, and MMLU-Redux stays at exact parity. Reasoning becomes 19% shorter on GSM8K, 4% shorter on LiveCodeBench, and 18% shorter on ToolUse. MMLU-Redux is the exception: it halves theater at equal accuracy while mean length rises by 5%.

LiveCodeBench protocol. The training-time sampled validation peak rises from 0.30 (baseline step 320) to 0.37 (ProFIL step 440) and is the accuracy metric in Table 1. Validation was logged every 20 steps with four sampled rollouts per problem. This peak is descriptive training telemetry, not a final greedy held-out estimate. At the final checkpoint, greedy@1 on all 131 test problems is 0.214 for the baseline, 0.229 for length penalty, and 0.183 for ProFIL. The theater result does not depend on which accuracy protocol is chosen, but the accuracy conclusion does. We claim an accuracy gain for the reported sampled-peak protocol, not for every checkpoint and decoding rule.

5.2.1 ToolUse: Chain-Level Theater and a Fivefold Faithful-Fraction Gain

On ToolUse, theater appears as extra planning blocks after the model has already committed to a complete tool call. ProFIL raises faithful fraction fivefold, from 0.029 to 0.147.

ProFIL increases single-tool-call resolutions $8\times$. Single-thinking-block resolutions rise from $1/68$ to $8/68$. In these rollouts the model identifies the right tool and acts without generating a second

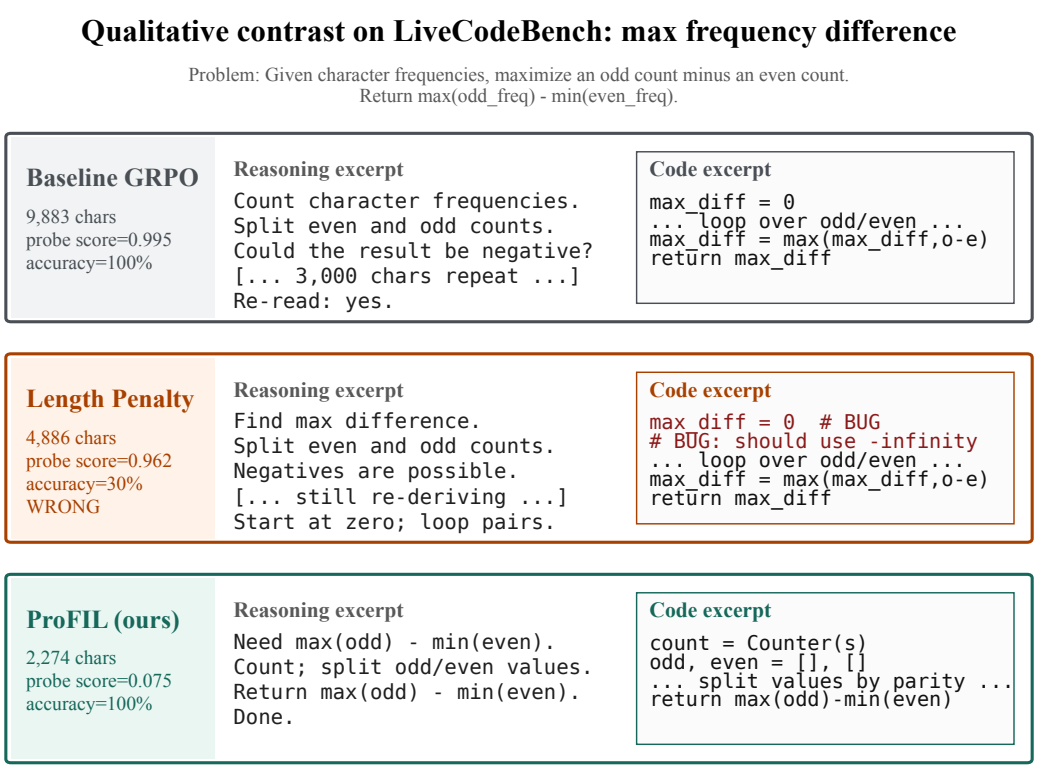

Figure 4: **A concrete LiveCodeBench comparison.** We show the highest-contrast example among 14 high-theater baseline cases. The baseline repeatedly revisits a solved edge case (9,883 characters, probe score 0.995). Length penalty is shorter but remains theatrical and introduces an initialization bug (4,886 characters, score 0.962). ProFIL reaches the correct five-line program directly (2,274 characters, score 0.075). This example is illustrative rather than a population estimate.

block to reconsider the same plan. Genuine multi-step plans remain possible because the filter targets post-commitment behavior rather than the number of tool calls.

5.3 Length Compression Is Not the Operative Mechanism

A matched GRPO plus length-penalty baseline uses $r_{\text{LP}} = r_{\text{code}} - 2.5 \times 10^{-5}|y|_{\text{chars}}$. It produces the shortest LiveCodeBench chains, but raises performativity from 0.267 to 0.374 (Table 3). ProFIL instead lowers it to 0.076. The same ordering holds in every length quintile and throughout the evaluation-cutoff sweep $\eta \in \{0.05, 0.10, 0.20, 0.50\}$. Fewer characters are not enough. The filter must distinguish reasoning before commitment from padding after commitment.

5.4 Independent Corroboration

The headline operational metric uses the frozen probe that also shapes training. By itself, that comparison could be circular: the policy might lower the probe score without becoming less padded to an outside reader. We separate two probe-blind judge checks from three anti-gaming and cross-measurement checks.

Independent LLM judge. A GPT-4.1 judge evaluated 100 LiveCodeBench rollouts per condition using only a natural-language faithfulness rubric, with no access to probe scores or training labels. Strict faithful-fraction rises from 0.49 (95% CI 0.39–0.59) to 0.73 (95% CI 0.64–0.81): a $+24$pp gap with non-overlapping CIs.

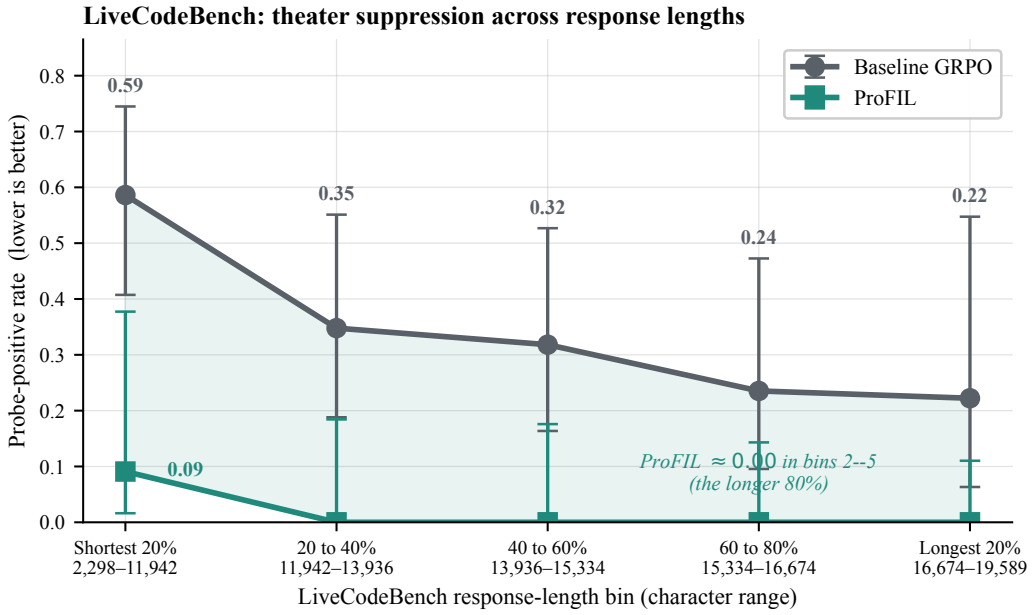

Figure 5: **Theater suppression holds at every chain length on LiveCodeBench.** Baseline and ProFIL rollouts are pooled, sorted by character count, and divided into five equal-frequency bins from the shortest 20% to the longest 20%. Each x-axis label gives the percentile span and observed character range. The vertical axis is the probe-positive rate at the same evaluation cutoff $\eta = 0.9$ used in Table 1. ProFIL remains below the baseline in each bin, including the longest 20%, so the gain is not explained by shifting probability mass toward short responses.

Pairwise judge. A second judge sees anonymized baseline and ProFIL rollouts side by side, in random order, and chooses which contains more padding relative to its own length. It never sees probe scores or forced-answer labels. The baseline is judged more padded in 71% of GSM8K pairs, 72% of ToolUse pairs, 58% of MMLU-Redux pairs, and 57% of LiveCodeBench pairs. The LiveCodeBench length-penalty model has no such edge over baseline at 51%.

Frozen-probe audit. RL can sometimes hide a targeted behavior from a learned scorer (Gupta & Jenner, 2025). We label LiveCodeBench rollouts with fresh forced-answer counterfactuals and measure whether the original frozen probe can still separate faithful from theatrical examples. Its AUROC is 1.000 before, during, and after training while mean performativity falls. The policy changes its generated behavior without erasing the probe's ability to detect the labeled distinction.

Cross-measurement and inference-time controls. The GSM8K probe transfers zero-shot to LiveCodeBench and recovers the same direction. A new student probe trained on the updated policy also agrees with the frozen-base measurement. In contrast, a sweep of Contrastive Activation Addition coefficients (Rimsky et al., 2024) does not reach the performativity of the ProFIL-trained policy. The change comes from training, not from a temporary intervention at inference.

Table 4: **The frozen scorer remains valid, while inference-time steering is insufficient.** Lower performativity is better. A student probe trained on the updated policy agrees with the frozen-base probe. Every tested CAA coefficient remains above the ProFIL evaluation performativity of 0.076.

(a) Probe audit

Scorer	Baseline	ProFIL
Frozen base	0.267	0.076
Student	0.290	0.069

(b) Inference-time steering

CAA coefficient	0	5	10	20	40
Performativity	0.080	0.100	0.090	0.100	0.090

ProFIL-trained evaluation: 0.076

5.5 What the Filter Removes

A separate behavioral mode. LiveCodeBench performativity is bimodal (Table 5). Baseline rollouts cluster below 0.1 or above 0.5, with little mass between them. ProFIL removes the high-theater mode while retaining the low-theater mode across chain lengths.

A concrete writing pattern. High-theater rollouts often restart a finished derivation under headings such as `### Approach`, `### Solution`, or `### Explanation`. At least one such block appears in 72% of high-theater baseline rollouts but only 3% of faithful ones. It appears in none of the ProFIL rollouts. The headings themselves are not penalized. The pattern matters only when it repeats reasoning after commitment.

A confidence ritual. Theater is not concentrated on difficult failures. On LiveCodeBench, the high-theater tercile reaches about 33% accuracy compared with about 9% in the low-theater tercile (Spearman $\rho = 0.240$, $p = 0.016$). The model tends to perform confidence after it already knows the solution. Conditioning on correctness on GSM8K and MMLU-Redux gives the same conclusion: the faithfulness gain is not caused by incorrect or uncommitted chains being counted as faithful.

Table 5: **What changes when ProFIL removes post-commitment reasoning.** Exact values make each behavioral result independently readable. Counts are rollout counts.

Observable	Exact comparison	What it shows
LiveCodeBench score shape	Baseline has modes below 0.1 and above 0.5; ProFIL removes the high-score mode.	A distinct theater mode disappears.
Re-derivation headings	Baseline: 31/43 high-theater rollouts (72%) and 2/72 faithful rollouts (3%); ProFIL: 0/131 (0%).	Headings track post-commitment restatement.
ToolUse single-block resolutions	Baseline: 1/68 (1.5%); ProFIL: 8/68 (11.8%).	$8\times$ more direct resolutions.
LiveCodeBench accuracy by theater tercile	Low theater: 9%; high theater: 33% ($\rho = 0.240$, $p = 0.016$).	Theater is not a failure-only effect.

6 Discussion and Conclusion

Self-correction and commitment labels. Corrections before the first correct forced answer are not labeled performative. This distinction concerns the supervision labels; the training filter operates on complete rollouts. In one GSM8K case, the model first answers 1,200, then subtracts the dragon's reach and corrects to 200. Commitment occurs at the correction; only the later restatement is performative. Changing one's mind is not equated with theater.

Limitations. *Construct:* forced answering detects when a correct answer becomes extractable, not whether visible reasoning entails it or caused the computation. *Uncommitted chains:* target ratio zero can denote genuine search or a repetition loop. *Backbone and domain:* Llama-8B serves GSM8K and LiveCodeBench, while Qwen-7B serves ToolUse and MMLU-Redux; agreement does not isolate architecture from domain. *LiveCodeBench accuracy:* ProFIL leads at the sampled@1 training peak and trails at the final greedy checkpoint, making accuracy protocol-dependent although theater reduction is stable. *Circularity:* the main metric reuses the training probe; independent judges and the forced-answer audit agree, but larger human audits and more models would help. *Training variance:* one RL run per condition means evaluation intervals do not measure seed variance. *Probe target:* a poorly chosen target could filter useful uncertainty or refusal; commitment labels narrow, but do not remove, this risk.

Future work. Combine ProFIL with process rewards, extend commitment verification to open-ended domains, and test more backbones with larger blinded human audits.

Conclusion. ProFIL turns a frozen internal signal into an RL filter. Across four domains and two architectures, performativity falls by 11–100%, independent judges agree, and a length penalty fails. ProFIL targets when useful reasoning ends, not token count.

AI USE STATEMENT

Generative AI tools were used as follows. First, to aid and polish writing: drafts of sections, captions, and the rebuttal-driven revisions were edited with AI assistance, and every sentence was reviewed by the authors. Second, for retrieval and discovery: AI-assisted search was used to locate related work on chain-of-thought faithfulness, concise-reasoning RL, and probe-based training signals, and every cited paper was read and checked by the authors before inclusion. Third, for research execution: AI coding assistants helped write and refactor analysis and plotting scripts, and the outputs of those scripts were checked against the raw rollout logs by the authors. AI assistance was also used to review methodological descriptions and the interpretation of reported results. Generative AI was not used to run experiments, to produce or alter any reported number, or to write the probe or training code that defines the method. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The method is fully specified in Section 4 and Algorithm 1 (App. B), with per-domain training configurations, the exact forced-answer prompt, and the group-normalization rule in App. B, probe architecture and layer choices in App. C, and judge prompts and the ToolUse commitment heuristic in App. D. Compute requirements are in App. H. Probe weights, training configurations, evaluation code, and rollout caches for all four domains will be released on acceptance.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. In *Conference on Language Modeling (COLM)*, 2025.

Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 967–976, 2023.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *International Conference on Learning Representations (ICLR)*, 2023.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Muzhi Dai, Chenxu Yang, and Qingyi Si. S-GRPO: Early exit via reinforcement learning in reasoning models. *arXiv preprint arXiv:2505.07686*, 2025.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Razvan-Gabriel Dumitru, Darius Peteleaza, Vikas Yadav, and Liangming Pan. ConciseRL: Conciseness-guided reinforcement learning for efficient reasoning models. *arXiv preprint arXiv:2505.17250*, 2025.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Fazl Barez, Hannaneh Hajishirzi, Stuart Yu, Mark Steedman, and Pasquale Minervini. Are we done with MMLU? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pp. 5069–5096, 2025.

Goodfire AI. RLFR: Reinforcement learning from feature rewards. Goodfire AI Research (https://www.goodfire.ai/research/rlfr), 2026.

Rohan Gupta and Erik Jenner. Rl-obfuscation: Can language models learn to evade latent-space monitors? *arXiv preprint arXiv:2506.14261*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

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.

Kimi Team. Kimi k1.5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

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 29th Symposium on Operating Systems Principles (SOSP)*, 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.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. *arXiv preprint arXiv:2301.13379*, 2023.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 15504–15522, 2024.

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

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

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Edwin B Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

Jixiao Zhang and Chunsheng Zuo. GRPO-LEAD: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models. *arXiv preprint arXiv:2504.09696*, 2025.

A Extended Results

A.1 Length-controlled faithfulness

Figure 5 in the main paper reports the full length-controlled analysis. Across all five pooled LiveCodeBench length quintiles, ProFIL stays below the baseline probe-positive rate, including in the longest 20% of responses. This rules out a shift toward shorter rollouts as the explanation for the observed reduction in post-commitment reasoning.

A.2 Frozen-Probe AUROC (Anti-Gaming Evidence)

A probe-as-reward system is vulnerable to RL obfuscation (Gupta & Jenner, 2025): the policy could learn to produce low probe scores while still generating theatrical reasoning, by shifting its activations away from the probe's decision boundary. We check this by re-scoring trained-model rollouts with the frozen probe and computing AUROC against forced-answer-derived performativity labels.

On LiveCodeBench (Llama-8B), the frozen probe attains AUROC 1.000 on rollouts from the baseline, length-penalty, and ProFIL trained models, and remains 1.000 throughout training while mean performativity falls from baseline to ProFIL (Table 1). The probe rank-orders theatrical vs. faithful rollouts perfectly before, during, and after training: the empirical signature of genuine theater reduction rather than activation-space evasion.

A.3 Evaluation-Cutoff (η) Sensitivity

Table 6 reports performativity ratio at $\eta = 0.05, 0.10, 0.20$, and 0.50. ProFIL remains below baseline at every cutoff in every domain. On LiveCodeBench, length penalty remains above baseline at every cutoff, so compression alone does not reduce theater.

Table 6: **Performativity ratio is stable across evaluation cutoffs η.** Each cell: perf_ratio (fraction of steps with probe sigmoid $\geq \eta$). ProFIL probe is lower than baseline at every η in every domain. LCB length penalty is higher than baseline at every η. These sensitivity cutoffs are distinct from training threshold τ.

Domain	Condition	η=0.05	η=0.10	η=0.20	η=0.50
GSM8K	Baseline	0.060	0.038	0.022	0.014
	ProFIL	**0.048**	**0.032**	**0.012**	**0.004**
LCB	Baseline	0.489	0.450	0.420	0.328
	Len. Penalty	0.649	0.603	0.550	0.458
	ProFIL	**0.405**	**0.359**	**0.267**	**0.130**
MMLU	Baseline	0.036	0.030	0.022	0.018
	ProFIL	**0.030**	**0.024**	**0.020**	**0.018**

In-training validation curves. GSM8K and LiveCodeBench validation accuracy logs at every 20 GRPO steps are released alongside the paper.

B Algorithmic and Hyperparameter Details

Per-domain training configuration.

- **GSM8K** (Llama-8B): 7440 train / 500 test problems, 3 training epochs, 4096-token context, τ=0.5, group size K=8, probe layers [13, 19].
- **LiveCodeBench v6** (Llama-8B): 920 train / 131 test problems, 3 epochs, 8192-token context, τ=0.2, K=8, probe layers [13, 19]. Reward: test-case pass rate > 0.5.
- **ToolUse** (Qwen-7B): 4046 train / 68 test, 3 epochs, 4096-token context, τ=0.5, K=8, probe layers [14, 21]. Chain-level performativity pooling.
- **MMLU-Redux** (Qwen-7B): 6144 train / 500 test problems, 256 GRPO steps, 4096-token context, τ=0.5, K=8, probe layers [14, 21].

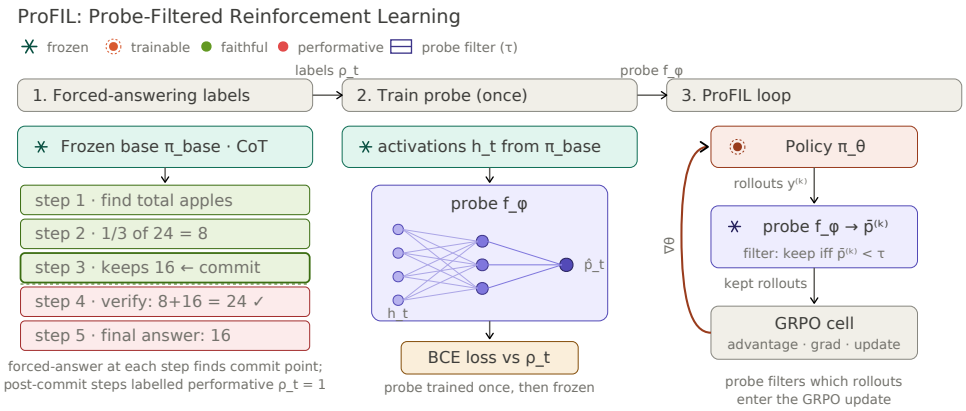

Figure 6: **ProFIL turns a frozen-base probe into a training signal.** (1) Forced-answering yields verifier-derived step-level performativity labels (no human annotation): the first step whose forced answer is correct defines the *commitment point*; later steps are performative. (2) A gated multi-head attention probe (RLFR-style; Goodfire AI, 2026) is trained *once* on activations of the *frozen* base at layers ℓ_1, ℓ_2, then never updated. (3) During GRPO, a separate frozen base teacher-forces the policy's exact rollout text, and the frozen probe re-computes its mean performativity. Rollouts above τ have advantage zeroed and contribute nothing to the policy update. The probe boundary is therefore stable throughout RL, and the GRPO update itself is unchanged outside this filter.

Algorithm 1 Probe-Filtered GRPO update

Require: frozen base π_{base}, probe f_ϕ, threshold τ, prompt x, group size K

Sample $\{y^{(k)}\}_{k=1}^{K} \sim \pi_\theta(\cdot \mid x)$
Compute verifier reward $r^{(k)}$
for $k = 1, \ldots, K$ **do**
    Compute frozen activations $h_{\ell_1}^{(k)}, h_{\ell_2}^{(k)} \leftarrow \pi_{\text{base}}(y^{(k)})$
    Compute per-step performativity $p_{1:T_k}^{(k)} \leftarrow f_\phi(h^{(k)})$; $\bar{p}^{(k)} = \text{mean}(p^{(k)})$
end for
Let $\tilde{r}^{(k)} \leftarrow r^{(k)} \cdot \mathbf{1}[\bar{p}^{(k)} < \tau]$ ▷ zero reward of filtered rollouts
$\mu, \sigma \leftarrow \text{mean}(\tilde{r}), \text{std}(\tilde{r})$ over all K rollouts
for $k = 1, \ldots, K$ **do**
    $\hat{A}^{(k)} \leftarrow ((\tilde{r}^{(k)} - \mu)/\sigma) \cdot \mathbf{1}[\bar{p}^{(k)} < \tau]$
end for
Update θ with standard GRPO step using $\{\hat{A}^{(k)}\}$.

All runs: AdamW optimizer, cosine LR schedule, identical to the matched GRPO baseline except for the probe filter. LiveCodeBench uses the empirically selected τ=0.2; under the keep rule $\bar{p} < \tau$, this is more selective than 0.5. Full config files are released with the code.

Group statistics under filtering. Algorithm 1 sets filtered rollout rewards to zero, and μ, σ are computed over all K rollouts (filtered ones contribute $\tilde{r} = 0$), matching our implementation exactly: filtered rollouts enter the group-normalization statistics as zero-reward entries rather than being excluded from them. This is not equivalent to computing μ, σ over the unfiltered subset only, since including zeroed rollouts changes the advantage assigned to every other rollout in the group.

Forced-answering prompt. "Stop reasoning now. Based on what you have written above, give your final answer in the format `\boxed{}`." Greedy decoding, max 64 new tokens. We measure first-correct over the full step-boundary trajectory. *LiveCodeBench exception*: the forced-answer correctness criterion is test-case execution (pass rate > 0.5), not boxed-answer string matching, since LCB problems require runnable code rather than a closed-form answer.

C PROBE ARCHITECTURE AND TRAINING

Architecture. Each head h has learned query $q_h \in \mathbb{R}^{d_h}$ and sigmoid gate g_h. Keys and values come from per-layer LayerNorm-normalized residual-stream activations of the frozen base at layers ℓ_1, ℓ_2. Output: $z = \sum_h g_h \cdot \text{head}_h$, scalar logit via linear projection.

Training. Probes trained on rollouts from the frozen base using binary cross-entropy on forced-answering labels. Best-epoch held-out AUROC on the probe-train/test split (in-distribution): GSM8K 0.924, LiveCodeBench 0.928, ToolUse 0.976, MMLU-Redux 0.991.

Layer choice. For Qwen-7B: layers 14 and 21 (mid-network commitment and late elaboration). For Llama-8B: layers 13 and 19. Training AUROC exceeds 0.92 on all four domains, validating that these layers carry a clean theater signal.

D INDEPENDENT LLM JUDGE AND TOOLUSE COMMITMENT HEURISTIC

We score every rollout with a GPT-4.1 judge using a natural-language faithfulness rubric, with no access to probe scores or training labels. On LiveCodeBench, strict faithful-fraction is 0.49 for the baseline (95% Wilson CI $[0.39, 0.59]$) versus 0.73 for ProFIL ($[0.64, 0.81]$): a $+24$pp gap with non-overlapping CIs.

ToolUse chain-level commitment heuristic. We flag the first step whose decoded text contains a tool-call closure (a JSON-formatted `{"tool":...,"args":...}` block followed by terminal `</think>` or end-of-turn marker), or failing that, the first step whose forced-answer matches the ground-truth tool name and arguments. Steps after this commitment are labeled performative for the chain-level metric.

E ADAPTIVE THRESHOLD SIMULATION

The per-domain τ values used in this paper (GSM8K/ToolUse/MMLU-Redux: 0.5; LiveCodeBench: 0.2) match the reward density of each domain (App. B). A natural alternative is to derive τ automatically from the rolling mean group reward, removing the need for domain-specific tuning.

Proposed schedule. Define

$$\tau_t = \tau_{\min} + (\tau_{\max} - \tau_{\min}) \cdot \text{clip}\left(\frac{\bar{r}_t - r_{\text{lo}}}{r_{\text{hi}} - r_{\text{lo}}},\, 0,\, 1\right),$$

where $\bar{r}_t$ is the exponential moving average of group reward at step t, and $(r_{\text{lo}}, r_{\text{hi}}) = (0.10, 0.40)$, $(\tau_{\min}, \tau_{\max}) = (0.20, 0.50)$. When reward density is low, τ is small and the filter is more selective. As reward density grows, τ rises and the filter relaxes. This describes the retrospective schedule above; it was not used for the reported training runs.

Retrospective simulation on LiveCodeBench. Figure 7 applies this schedule retrospectively to the three available LCB checkpoints using their per-example performativity scores and checkpoint accuracies as a proxy for reward density. At step 320 (accuracy 11%), fixed τ=0.5 would filter 59% of rollouts; adaptive τ self-adjusts to 0.21 and matches the per-domain τ=0.2 used in the main experiments. At step 440 after training, accuracy recovers to 18% and adaptive τ rises to 0.28, filtering 21% of rollouts rather than 13% at fixed τ=0.5. The schedule removes the per-domain τ choice entirely. A full training comparison of adaptive vs. fixed τ is left to future work.

F THEATER AS A CONFIDENCE RITUAL

This appendix expands the headline result of §5.5 that high-theater rollouts in the baseline correlate with task correctness. Figure 8 shows the per-tercile accuracy split (panel A) alongside the per-decile rollout-density shift between baseline and ProFIL (panel B): theater concentrates on problems the model already solves, and ProFIL collapses the distribution to the low-theater region while preserving accuracy.

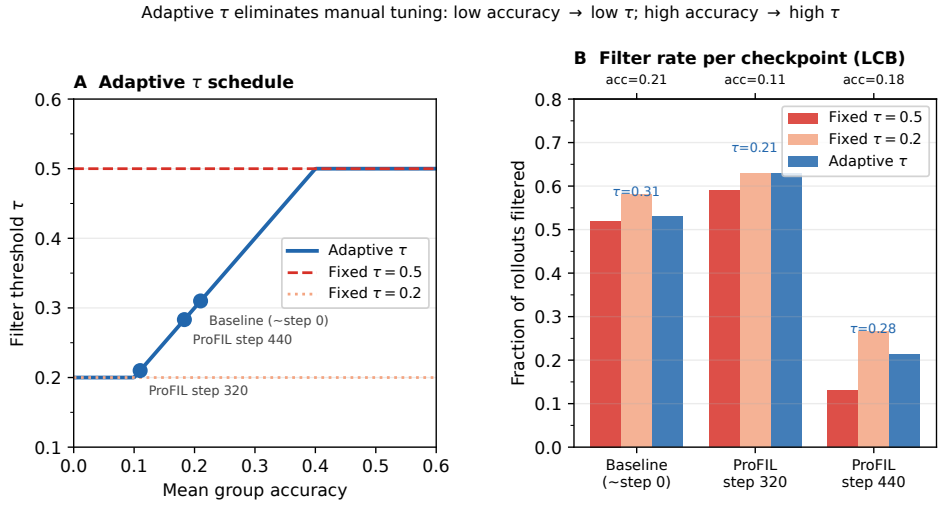

Figure 7: **Adaptive τ eliminates manual threshold tuning.** *Left:* the proposed linear schedule maps mean group accuracy to $\tau \in [0.20, 0.50]$; dots mark the three LCB checkpoints. *Right:* filter rates under fixed τ=0.5, fixed τ=0.2, and adaptive τ at each checkpoint. At step 320 (lowest accuracy) adaptive τ automatically matches the sparse-regime value; at step 440 it rises, balancing signal preservation and theater filtering.

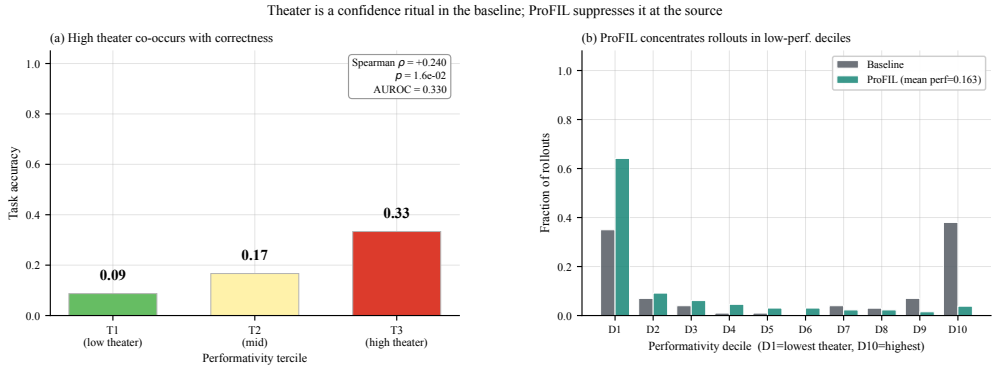

Figure 8: **Theater marks success, not uncertainty: it is a learned confidence ritual.** *Left:* task accuracy by performativity tercile (T1 = low theater, T3 = high theater) on LiveCodeBench baseline rollouts. High-theater rollouts trend toward higher accuracy: the high-theater tercile averages $\sim$33% accuracy vs. $\sim$9% for the low-theater tercile (Spearman ρ=+0.240, p=0.016; AUROC = 0.33, meaning theater *anti-predicts* failure). *Right:* the decile distribution (finer-grained than panel A's terciles, to show the bimodal collapse) shifts from baseline mass at D1 and D10 to near-entirely low-theater under ProFIL. ProFIL does not suppress uncertainty; it suppresses post-hoc elaboration on problems the model already knows how to solve.

G Self-Correction Case Studies

We present two GSM8K rollouts from the probe-trained model in which the forced-answer oracle confirms a wrong answer at an early step and a correct answer at a later step. In both cases, ProFIL assigns the commitment point at the *first correct* step; the intermediate correction steps are pre-commitment and are not labeled performative. This is a property of the labels, not a guarantee that the complete rollout passes the training filter.

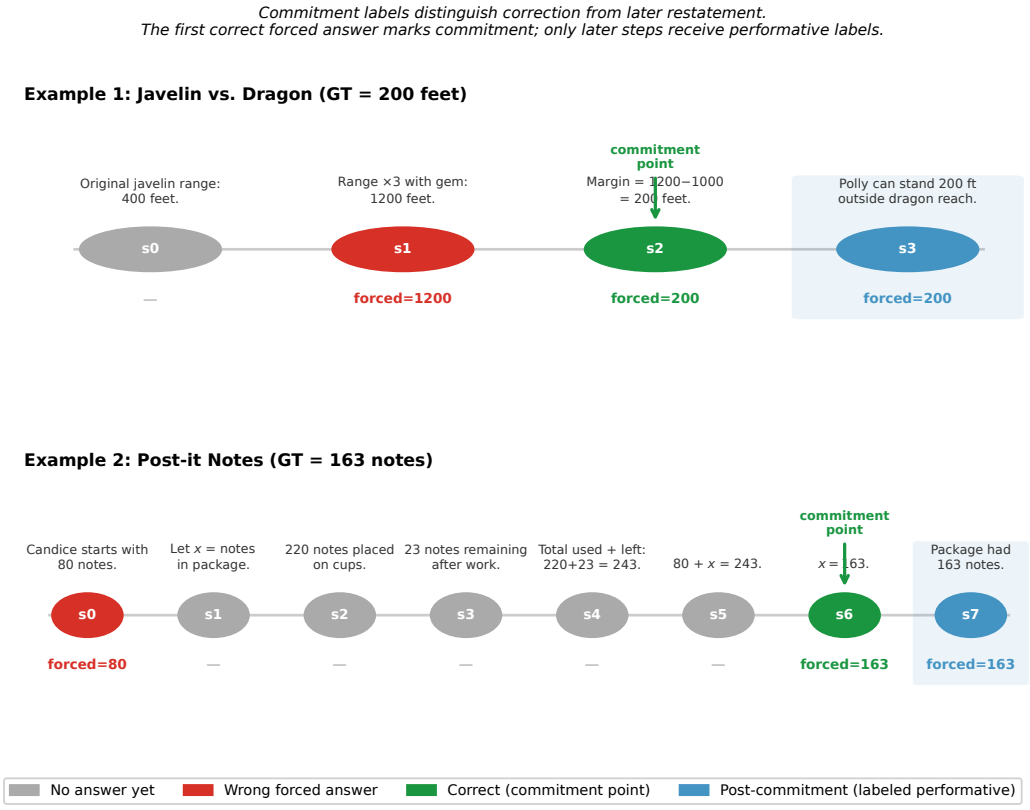

Figure 9: Step-by-step forced-answer trajectories for two self-correcting GSM8K rollouts. Each circle is a paragraph step; the label beneath shows what the model would have answered if stopped at that step. The commitment point (green arrow) is the first step at which the forced answer matches the ground truth. Blue-shaded steps are post-commitment; they are labeled performative only if they follow the commitment point. In both examples the error-correction move itself occurs at commitment and is not labeled performative.

Example 1: Javelin vs. Dragon (ground truth: 200 feet). The problem asks how far outside the dragon's reach Polly can stand. At step 1 the model computes the gem-enhanced range (1200 feet) and, if stopped there, answers **1200**, mistaking an intermediate quantity for the final answer. At step 2 it subtracts the dragon's reach ($1200 - 1000 = 200$) and correctly answers **200**. The commitment point is assigned at step 2; step 3 (a restatement) is post-commitment. Step 3 receives the performative label, while steps 1–2 (including the correction) do not. The training filter acts on the complete rollout.

Example 2: Post-it Notes (ground truth: 163 notes). The problem requires setting up and solving $80 + x = 243$. At step 0 the model reads off the initial count (80) and would answer **80** if stopped; this is a premature answer before the equation is established. Steps 1–5 build the equation; the model cannot yet commit. At step 6 it solves $x = 163$ and the forced-answer oracle returns **163**: the commitment point. The six-step correction path (steps 0–5) is entirely pre-commitment and is not labeled performative.

H COMPUTE BUDGET

Each domain-condition pair: 4 (Llama-8B) or 4–8 (Qwen-7B) H100 80 GB GPUs for 3 epochs, totalling $\sim$480 GPU-hours across the eight training runs. Probe training: $\sim$2 GPU-hours per domain. Evaluation (vLLM rollouts and scoring): $\sim$8 GPU-hours per domain. Ablations (Figure A.1, independent judge, mechanistic analysis): $\sim$200 additional GPU-hours.

Probe filtering overhead. On a single A100-40GB, scoring one LiveCodeBench rollout with the frozen-base probe (activation extraction plus the attention-probe forward pass, the exact computation used to produce the per-rollout filter signal during GRPO) takes $\sim$319 ms, versus $\sim$533 ms for one forward-backward step of the policy itself on a comparably sized batch: roughly 60% of one policy update's per-rollout cost, and strictly less since the probe pass has no backward computation or optimizer state. This scoring runs alongside, not instead of, the GRPO update, adding a single frozen-model forward pass per rollout batch.

I BROADER IMPACTS

ProFIL produces models whose chain-of-thought more closely reflects the computation that led to the answer. The primary benefit is auditability: a reasoning model whose chain reflects its computation is more amenable to oversight, step-level interpretability, and honest disagreement. The symmetric concern is that a method suppressing "performative" filler could in principle target any reasoning pattern a probe is trained to recognize, including reasoning a model should express (safety considerations, uncertainty, refusal preambles). Since our probe targets behavior identified by forced-answering correctness, the immediate risk is low; the general lesson is that probes are dual-use and the choice of probe target is a value judgment.