

Anti-Prior Reinforcement Learning for Context-Grounded Reasoning

Zheng Wang¹ Zexin Lu² Xuming Hu^{1,3†}

¹The Hong Kong University of Science and Technology (Guangzhou)

²The Hong Kong Polytechnic University

³The Hong Kong University of Science and Technology

zhengwang.0426@gmail.com xuminghu@hkust-gz.edu.cn

Abstract

Large language models are increasingly asked to treat the input context as the source of task-specific rules, procedures, and evidence. A difficult case arises when this context contradicts pretrained priors: models may mention the local rule but still execute familiar semantics, commonsense, positional, or operator shortcuts. We study this failure mode as *prior-dominated reasoning*, where either the final answer or the intermediate trajectory follows a plausible prior-consistent path rather than the current context. We propose *Anti-Prior Reinforcement Learning*, a post-training framework that constructs executable context-prior conflict tasks and trains with a gated verifier reward. The reward gives positive credit for answer correctness, context anchoring, and step-wise local rule use, while penalizing prior-consistent answers and reasoning steps. On Qwen3-8B, Anti-Prior RL improves controlled validation accuracy from 51.7% to 74.5% and controlled OOD accuracy from 46.4% to 73.4%, while reducing OOD prior-answer/prior-step hit rates from 15.9%/40.2% to 1.5%/8.9%. It also improves over SFT and LongRLVR on downstream CL-Bench, RULER-QA, and Long-Bench v2 across both Qwen3-8B and Qwen2.5-7B-1M, and outperforms outcome-only RL on Qwen3-8B. Together, these results provide complementary evidence that the learned behavior transfers beyond the constructed prior-conflict setting to broader context-learning and reasoning benchmarks.

1 Introduction

Long-context language models are increasingly expected to use the current input as both facts and task specification, following in-context learning (Brown et al., 2020; Min et al., 2022). In realistic settings, context may define local rules, temporary ontologies, default overrides, repository conven-

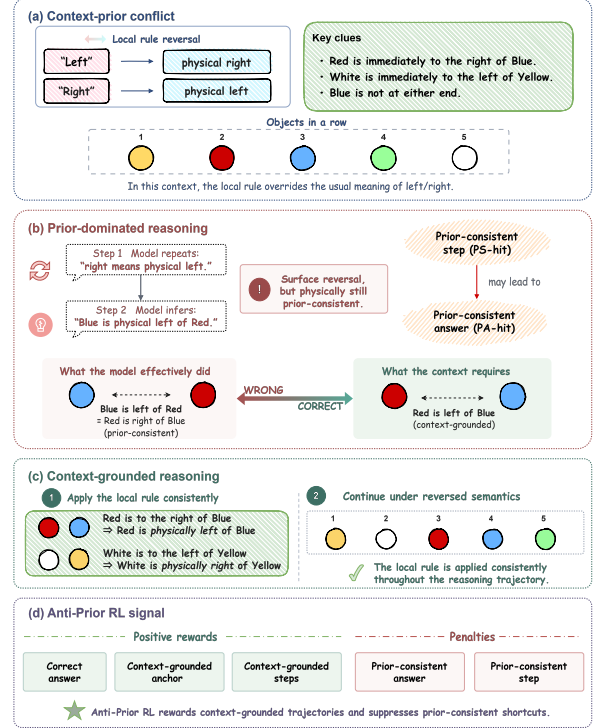


Figure 1: Prior-dominated reasoning under context-prior conflict. The context locally reverses the meanings of “left” and “right.” A model may state the rule correctly but infer “Blue is left of Red,” which is equivalent to the prior physical relation “Red is physically right of Blue.” Context-grounded reasoning instead applies the local rule consistently, and Anti-Prior RL rewards such trajectories while penalizing prior-consistent steps and answers.

tions, or procedures absent from pretraining. Recent context-learning and long-context evaluations show that models still struggle to apply such information reliably (Dou et al., 2026; Liu et al., 2024; Shi et al., 2023; Hsieh et al., 2024; Bai et al., 2025; Wu et al., 2025; Bertsch et al., 2025; Bai et al., 2024; Gao et al., 2025; Wang et al., 2025). This matters because many applications require treating context as binding rather than background information.

The most revealing case is not ordinary retrieval

[†]Corresponding author.

failure, but *context-prior conflict*: the context contradicts what the model would normally assume, echoing work on contextual-parametric knowledge conflicts and prompt-induced biases (Longpre et al., 2021; Zhao et al., 2021). Figure 1 gives a minimal example. The puzzle locally redefines “left” to mean physical right and “right” to mean physical left. A context-grounded model should apply this rule consistently, e.g., interpreting “Red is to the right of Blue” as Red being physically left of Blue. The target failure is subtler: a model may state that “right” means physical left, but then infer “Blue is left of Red.” This surface reversal preserves the usual prior relation—Red is physically right of Blue—and is therefore a prior-consistent intermediate step rather than context-grounded reasoning.

We study this failure mode as *prior-dominated reasoning*, related to shortcut learning and “right-for-the-wrong-reasons” behavior (Geirhos et al., 2020; McCoy et al., 2019; Gururangan et al., 2018; Jia and Liang, 2017). Operationally, a prior is a default semantic, commonsense, positional, evidential, or operator assumption that is plausible under pretraining and can be instantiated as a verifiable shortcut path. We do not inspect latent priors directly; instead, we use controlled cases where the context-defined and prior-consistent solutions make different predictions. A model is prior-dominated when its final answer or intermediate steps match the prior path despite contextual evidence for the gold answer.

This framing is useful because it distinguishes several behaviors that answer-only evaluation conflates (Cobbe et al., 2021; Uesato et al., 2022; Lightman et al., 2024; Lanham et al., 2023). A model may fail to find the relevant context; it may find the context but not apply it; or it may mention the context while still following a familiar prior path in its intermediate steps. The last case is particularly hard to diagnose from final answers alone, because prior-consistent intermediate states can appear even when the final answer is not an exact prior answer. We therefore evaluate not only accuracy, but also *prior-answer hit* and *prior-step hit*, which measure whether the completion matches known shortcut answers or known shortcut steps generated by the task program.

Existing methods only partially address this problem. Chain-of-thought prompting, self-consistency, scratchpads, rationale bootstrapping, decomposition, and tool use can elicit or aggregate traces (Wei et al., 2022; Wang et al., 2023;

Nye et al., 2021; Zelikman et al., 2022; Zhou et al., 2023; Yao et al., 2023), but longer traces do not ensure that the model follows the local rule (Lanham et al., 2023). Long-context alignment and verifiable context rewards improve evidence use (Bai et al., 2024; Gao et al., 2025; Zhang et al., 2025; Chen et al., 2026), but they do not explicitly distinguish a context-grounded trajectory from a prior-consistent trajectory. Supervised fine-tuning on anti-prior examples can expose the model to the desired behavior, but likelihood training alone does not directly penalize prior-consistent shortcuts sampled during generation.

We propose **Anti-Prior Reinforcement Learning**, a post-training framework for *observable, constructible, and verifiable* context-prior conflicts. It constructs controlled conflict tasks and trains against explicit shortcut paths that can be generated and checked by an executable task program. Each instance stores the rendered prompt, gold context-conditioned answer and path, plausible prior answers, prior-consistent intermediate states, and rule or evidence markers. The model only sees the rendered prompt. The verifier uses hidden program fields to reward answer correctness, context anchoring, and step-wise local rule application, while penalizing prior-answer and prior-step hits. This design turns context-prior conflict into a verifiable training signal: the model is not merely rewarded for being correct, but for being correct through the current context rather than through a shortcut. We do not claim a universal mitigation procedure for arbitrary datasets where such conflict structure cannot be specified.

Our controlled data include seven task families: scoped document QA, query-focused meeting summarization, code/config dependency reasoning, few-shot rule transfer, uniqueness-constrained passage counting, ordering/timeline reasoning, and structured table conditional reasoning. The validation split uses disjoint instances from the same family-level distributions as training. The controlled OOD split preserves the same high-level capabilities but changes templates, rule instantiations, evidence layouts, context structures, and shortcut patterns. This lets us ask whether the model learns the intended context-over-prior behavior rather than memorizing surface templates. We further test transfer on independently constructed benchmarks: CL-Bench, RULER-QA, and Long-Bench v2.

Experiments show that Anti-Prior RL reduces

the controlled failure mode and transfers beyond the generator. On Qwen3-8B, it improves controlled validation accuracy from 51.7% to 74.5% and controlled OOD accuracy from 46.4% to 73.4%, while reducing OOD prior-answer and prior-step hit rates from 15.9%/40.2% to 1.5%/8.9%. The same trend holds for Qwen2.5-7B-1M. Downstream, Anti-Prior RL improves over SFT and LongRLVR on CL-Bench, RULER-QA, and LongBench v2.

Our contributions are:

- We define *prior-dominated reasoning*, a measurable context-under-utilization failure mode where models follow prior-consistent paths even when local context contradicts them.
- We propose *Anti-Prior Reinforcement Learning*, a verifier-based diagnosis and training framework for observable and constructible context-prior conflicts, using executable task programs and a gated reward to favor context-grounded trajectories and suppress prior shortcuts.
- We show that Anti-Prior RL reduces prior reliance under controlled and independently judged evaluations, generalizes to controlled OOD shifts, and yields transfer gains on context-learning and long-context benchmarks beyond the synthetic generator.

2 Related Work

Context learning and long-context utilization.

Work on in-context learning and prompt sensitivity shows that LMs adapt from examples but inherit format and label biases (Brown et al., 2020; Min et al., 2022; Zhao et al., 2021). Long-context LLMs also underuse available information: they are sensitive to evidence position (Liu et al., 2024), can be distracted by irrelevant context (Shi et al., 2023), and may over-rely on parametric knowledge under contextual conflict (Longpre et al., 2021). RULER, LongBench v2, LIFBench, and CL-Bench evaluate retrieval, aggregation, long-context reasoning, instruction following, and context learning (Hsieh et al., 2024; Bai et al., 2025; Wu et al., 2025; Bertsch et al., 2025; Dou et al., 2026); long-context alignment and training recipes motivate careful post-training for context use (Bai et al., 2024; Gao et al., 2025; Wang et al., 2025). Our goal is different: we construct training environ-

ments where context-defined and prior-consistent solutions make different, verifiable predictions.

Reasoning traces, grounding, and verifiable rewards. Prior work improves reasoning by eliciting or supervising intermediate steps: chain-of-thought prompting, scratchpads, rationale bootstrapping, least-to-most decomposition, self-consistency, and ReAct-style prompting encourage, supervise, or aggregate reasoning paths (Wei et al., 2022; Nye et al., 2021; Zelikman et al., 2022; Zhou et al., 2023; Wang et al., 2023; Yao et al., 2023), while attribution methods encourage grounding in evidence (Li et al., 2024). Process supervision and learned verifiers show that step-level feedback can outperform outcome-only supervision (Cobbe et al., 2021; Uesato et al., 2022; Lightman et al., 2024). Instruction tuning, preference optimization, and RLHF improve general instruction following (Christiano et al., 2017; Ziegler et al., 2019; Stienon et al., 2020; Ouyang et al., 2022; Schulman et al., 2017; Rafailov et al., 2023; Bai et al., 2022), but can leave context-specific failures (Wang et al., 2025). LongReward and LongRLVR add verifiable context rewards (Zhang et al., 2025; Chen et al., 2026), and recent reasoning-oriented RL systems provide related optimization context (Shao et al., 2024; DeepSeek-AI, 2025; Yu et al., 2025). Anti-Prior RL targets the narrower case where relevant context is present but the model follows a prior-consistent path.

3 Anti-Prior Reinforcement Learning

3.1 Problem Formulation

We study long-context reasoning tasks in which a model receives a prompt $x = (c, q)$ consisting of a context c and a query q . The context contains task-specific information such as local rules, definitions, evidence, procedures, or relational structures. The model produces a completion y containing a reasoning trace and a final answer. A standard answer-only training objective rewards y if the final answer matches the target answer. However, this objective cannot distinguish whether the answer was obtained by following the current context or by relying on a prior-consistent shortcut.

We define a *context-prior conflict* as a setting where the context-conditioned solution differs from a plausible solution induced by default semantic, commonsense, positional, or operator assumptions. Let a^* denote the gold answer obtained by applying the local context, and let $\mathcal{A}_p$ denote a set of

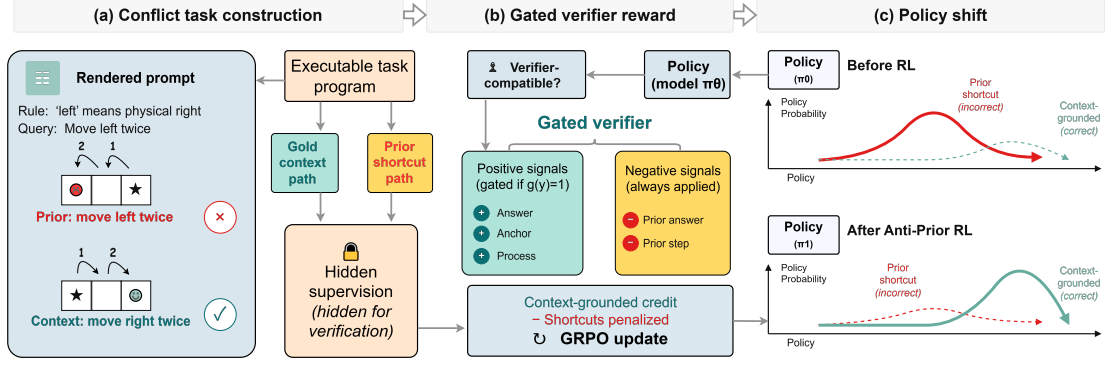


Figure 2: Overview of Anti-Prior RL. (a) An executable task program renders the prompt visible to the policy and stores gold context paths and prior shortcut paths as hidden verifier-only supervision. (b) Sampled completions are checked by a verifier-compatible gate $g(y)$; positive rewards for answer correctness, anchoring, and process steps are gated, while prior-answer and prior-step penalties suppress shortcut behavior. (c) GRPO uses the verifier reward to shift policy mass from prior-consistent shortcuts toward context-grounded reasoning.

plausible but context-invalid prior answers. In addition to prior answers, each task program may define a set of prior-consistent intermediate states or constraints $\mathcal{S}_p$, which allows us to detect prior-dominated reasoning even when the final answer is not an exact prior-answer match. A model exhibits *prior-dominated reasoning* when its final answer matches $\mathcal{A}_p$ or its intermediate steps match $\mathcal{S}_p$ despite the presence of contextual evidence supporting a^* . Our goal is to train models to prefer context-grounded trajectories: trajectories that identify the relevant contextual source, apply it consistently over intermediate steps, and produce the context-defined answer.

3.2 Overview

Figure 2 summarizes Anti-Prior RL as a three-stage training pipeline. First, an executable task program constructs a conflict instance: the rendered prompt is visible to the policy, while the gold context-grounded path and prior shortcut path are stored as verifier-only supervision. Second, sampled completions are passed through a strict parser and a verifier-compatibility gate $g(y)$, producing positive signals for answer correctness, context anchoring, and step-wise rule use, together with penalties for prior-consistent answers and steps. Third, GRPO updates the policy with these verifier rewards, shifting probability mass away from prior-consistent shortcuts and toward context-grounded reasoning trajectories.

3.3 Anti-Prior Data Construction

As illustrated in Figure 2(a), we construct controlled context-prior conflict tasks from executable

programs. Each program samples a local rule or evidence structure, renders a natural-language prompt, and stores hidden verification fields: the gold answer, the context-conditioned path, plausible prior answers, prior-consistent intermediate states, and rule or evidence markers. Only the rendered prompt is shown to the policy; hidden fields are used by the verifier for reward computation, diagnostics, and validation.

Our generator covers seven task families, each designed so that a plausible default shortcut gives the wrong answer while the correct solution requires following local context. The prior shortcut varies by family: it may be an out-of-scope salient answer, a query-agnostic summary, a default dependency rule, a memorized rule pattern, duplicate counting, surface temporal order, or a default table-reading heuristic. We filter instances whose gold and prior paths are ambiguous, whose gold path does not reproduce the stored answer, or whose relevant rule or evidence is missing from the rendered prompt. Detailed family definitions, filtering checks, verifier scope, and controlled OOD shifts are provided in Appendices A and C.

3.4 Gated Verifier Reward

The reward separates context-grounded trajectories from prior-consistent shortcuts, as shown in Figure 2(b). For a prompt x and completion y , we define:

$$R(x, y) = \text{clip}\left(g(y) \cdot \min\left(R^+(x, y), C(x, y)\right) - P_{\text{prior}}(x, y), -1, 1\right). \quad (1)$$

Here $g(y)$ is a verifier-compatibility gate, $C(x, y)$ is an answer-conditioned cap, and P_{prior} penalizes known prior-consistent answers and steps. The gate is computed by a strict parser over the completion: $g(y) = 1$ when the output contains a parseable final answer and a verifier-compatible reasoning field, and $g(y) = 0$ otherwise. Thus the gate is not assumed to be always active; unparseable completions receive no positive reward, although detectable prior-answer or prior-step penalties can still be applied. The positive reward is:

$$R^+(x, y) = \lambda_a R_{\text{ans}} + \lambda_c R_{\text{anchor}} + \lambda_p R_{\text{proc}}. \quad (2)$$

R_{ans} rewards final-answer correctness under the local context, R_{anchor} rewards identifying the relevant contextual rule or evidence, and R_{proc} rewards step-wise application of the local rule. Concretely, R_{proc} compares the parsed intermediate states or rule applications against the program’s gold path, gives normalized partial credit for matched steps, and stops accumulating after the first context-inconsistent or prior-driven step. This makes process reward a local credit signal rather than a second final-answer reward. The prior penalty is:

$$P_{\text{prior}}(x, y) = \lambda_{pa} P_{\text{pa}} + \lambda_{ps} P_{\text{ps}}. \quad (3)$$

P_{pa} penalizes final answers that match context-invalid prior answers, while P_{ps} penalizes intermediate steps that use a prior shortcut as an active inference step. We use fixed weights $\lambda_a = 0.25$, $\lambda_c = 0.20$, $\lambda_p = 0.35$, $\lambda_{pa} = 0.25$, and $\lambda_{ps} = 0.35$ across all task families. The answer-conditioned cap limits positive reward to 0.30 when the final answer is wrong and to 0.10 when the final answer matches a known prior answer. Additional parser and logging details are provided in Appendix C.

3.5 Reinforcement Learning Objective

We optimize the model using a GRPO-style objective with sequence-level verifier rewards (Schulman et al., 2017; Shao et al., 2024). As summarized in Figure 2(c), the update is intended to reallocate probability mass from prior-consistent trajectories to context-grounded ones. For each prompt x , the policy samples a group of K completions $\{y_1, \dots, y_K\}$. Each completion is scored by the verifier reward in Equation 1. We compute group-relative advantages from the normalized rewards and update the policy to increase the likelihood of completions with higher relative reward while

constraining deviation from the reference policy. Formally, the simplified objective is:

$$\mathcal{L}_{\text{GRPO}} = -\mathbb{E}_{x, \{y_i\}} \left[\frac{1}{K} \sum_{i=1}^K A_i \log \pi_{\theta}(y_i | x) \right] + \beta D_{\text{KL}}(\pi_{\theta} || \pi_{\text{ref}}). \quad (4)$$

where A_i is the group-relative advantage, π_{θ} is the trainable policy, π_{ref} is the reference policy, and β controls the KL penalty. Training and decoding details are reported in Appendix C; Appendix B provides an idealized analysis of how the reward suppresses prior-consistent trajectories and improves credit assignment.

4 Experiments

Our experiments ask whether Anti-Prior RL reduces prior-dominated reasoning, whether the behavior transfers beyond the generator, and which reward components drive the gains. We answer these with controlled prior-conflict evaluation, controlled OOD and downstream context-learning/long-context benchmarks, and grouped reward ablations over outcome reward, context anchoring, process supervision, and prior-shortcut penalties.

4.1 Experimental Settings

Models. We evaluate Anti-Prior RL on two base models: Qwen3-8B and Qwen2.5-7B-1M (Yang et al., 2025a,b). For downstream context-learning and long-context benchmarks, we compare the base model, supervised fine-tuning (SFT), LongRLVR, and our Anti-Prior RL model under the same evaluation protocol.

Training data. We train on 21K synthetic anti-prior prompts generated from the seven task families, with 3,000 training examples per family. Validation and controlled OOD evaluation each contain 1,400 examples, with 200 examples per family. The validation split uses disjoint instances sampled from the same family-level distributions as training. The controlled OOD split preserves the same high-level capability categories but holds out generation templates, rule instantiations, entity domains, evidence layouts, context structures, and shortcut patterns. Detailed task-program fields, conflict construction, dataset characterization, split construction, and filtering criteria are provided in Appendix A.

Baselines. We compare Anti-Prior RL against the base model, supervised fine-tuning (SFT), and LongRLVR. SFT tests whether exposure to anti-prior examples is sufficient without reinforcement learning. LongRLVR is the closest long-context RL baseline because it augments answer reward with verifiable context rewards. We implement LongRLVR under the same base models, LoRA configuration, rollout strategy, optimization budget, and evaluation protocol as Anti-Prior RL; the key difference is the reward formulation. LongRLVR rewards answer correctness and context utilization, whereas Anti-Prior RL additionally models trajectory-level prior reliance and penalizes prior-answer and prior-step shortcuts. This comparison therefore isolates the value of explicit shortcut modeling beyond generic verifiable context rewards. For reward ablations, we evaluate nested RL variants that add reward components in stages, allowing us to isolate the contributions of outcome reward, context anchoring, process supervision, and prior-shortcut penalties. To control for the larger sampled-completion budget used by RL, we also include *Reward-weighted SFT*, a same-rollout SFT control inspired by reward-weighted and ranked offline updates (Peters and Schaal, 2007; Peng et al., 2019; Dong et al., 2023; Yuan et al., 2023; Gulcehre et al., 2023). It starts from the SFT checkpoint, reuses the full rollout log collected during Anti-Prior RL, and trains with a supervised objective in which verifier-compatible completions are weighted by their verifier rewards rather than optimized with GRPO. This control gives SFT access to the same sampled completions and verifier scores, but it does not control the on-policy data-generation process.

Evaluation benchmarks. We evaluate on both controlled prior-conflict tests and common long-context/context-learning benchmarks. The controlled evaluation measures whether models follow context-defined rules under prior conflict and includes validation and controlled OOD splits. We report accuracy as the main metric, and additionally track verifier-derived diagnostics such as context anchoring, process correctness, and prior-hit when available. For downstream transfer, we evaluate on CL-Bench (Dou et al., 2026), RULER-QA (Hsieh et al., 2024), and LongBench v2 (Bai et al., 2025). CL-Bench evaluates whether models can learn new knowledge, rules, procedures, and empirical regularities from context; RULER-QA tests

long-context question answering under controlled context lengths; and LongBench v2 evaluates realistic long-context reasoning tasks. We follow official evaluation protocols and scoring scripts where available. To ensure fair comparison, all methods are evaluated with the same prompt format, maximum generation length, decoding configuration, and answer extraction rules within each benchmark. For controlled prior-conflict evaluations, we use the strict verifier parser to extract final answers and compute prior-hit diagnostics.

Training and inference details. We train Anti-Prior RL with a GRPO-style objective and LoRA adapters on Qwen3-8B and Qwen2.5-7B-1M. The default run uses 1600 GRPO update steps with 8 samples per prompt, yielding 25.6K prompt presentations and 204.8K sampled completions. Reward-weighted SFT reuses this rollout log as an offline pool and continues from the SFT checkpoint with reward-weighted supervised learning. Controlled evaluations use 1,400 validation and 1,400 controlled OOD examples with strict answer parsing. Full optimization and decoding details are in Appendix C; grouped reward ablations and Reward-weighted SFT report mean and standard deviation over three seeds.

4.2 Main Results

Controlled prior-conflict evaluation. Table 1 evaluates whether Anti-Prior RL directly reduces the target failure mode on our controlled prior-conflict validation and controlled OOD splits. On Qwen3-8B, Anti-Prior RL improves overall validation accuracy from 51.7% to 74.5% and controlled OOD accuracy from 46.4% to 73.4%. The improvement is accompanied by a substantial reduction in prior-consistent behavior: validation prior-answer hit drops from 16.1% to 1.6%, validation prior-step hit drops from 38.1% to 8.6%, and the same pattern holds on controlled OOD examples, where prior-answer and prior-step hit decrease from 15.9%/40.2% to 1.5%/8.9%. Gains are observed across all seven task families. The largest controlled OOD accuracy improvements occur on structured table conditional reasoning (+46.0), ordering/timeline query (+37.5), and meeting/query summary (+37.0), suggesting that the reward is especially effective when the correct solution depends on local rule definitions, temporal constraints, or query-specific evidence. On Qwen2.5-7B-1M, we observe the same trend: con-

Task family	Accuracy $\uparrow$				PA-hit $\downarrow$				PS-hit $\downarrow$			
	Val		OOD		Val		OOD		Val		OOD	
	Base	RL	Base	RL	Base	RL	Base	RL	Base	RL	Base	RL
Single-doc scoped QA	60.5	81.5 $\uparrow 21.0$	57.5	80.0 $\uparrow 22.5$	13.0	1.5 $\downarrow 11.5$	13.5	1.5 $\downarrow 12.0$	30.5	6.0 $\downarrow 24.5$	32.0	5.5 $\downarrow 26.5$
Meeting/query summary	40.5	79.5 $\uparrow 39.0$	43.0	80.0 $\uparrow 37.0$	21.5	1.0 $\downarrow 20.5$	17.0	0.5 $\downarrow 16.5$	47.5	6.0 $\downarrow 41.5$	44.0	5.0 $\downarrow 39.0$
Code/config dependency	61.5	72.5 $\uparrow 11.0$	54.5	74.0 $\uparrow 19.5$	11.5	2.0 $\downarrow 9.5$	12.0	1.5 $\downarrow 10.5$	28.0	9.5 $\downarrow 18.5$	31.0	7.0 $\downarrow 24.0$
Few-shot rule transfer	73.5	80.0 $\uparrow 6.5$	69.0	78.5 $\uparrow 9.5$	7.0	0.0 $\downarrow 7.0$	7.0	0.5 $\downarrow 6.5$	18.5	3.5 $\downarrow 15.0$	20.5	5.5 $\downarrow 15.0$
Passage count unique	42.5	59.0 $\uparrow 16.5$	40.0	57.5 $\uparrow 17.5$	20.5	4.5 $\downarrow 16.0$	20.0	3.0 $\downarrow 17.0$	48.5	18.5 $\downarrow 30.0$	49.0	20.5 $\downarrow 28.5$
Ordering/timeline query	46.0	81.0 $\uparrow 35.0$	41.5	79.0 $\uparrow 37.5$	17.5	1.0 $\downarrow 16.5$	19.0	1.5 $\downarrow 17.5$	43.5	5.0 $\downarrow 38.5$	46.0	6.0 $\downarrow 40.0$
Structured table conditional	37.5	68.0 $\uparrow 30.5$	19.0	65.0 $\uparrow 46.0$	21.5	1.5 $\downarrow 20.0$	23.0	2.0 $\downarrow 21.0$	50.5	11.5 $\downarrow 39.0$	59.0	13.0 $\downarrow 46.0$
Overall	51.7	74.5 $\uparrow 22.8$	46.4	73.4 $\uparrow 27.1$	16.1	1.6 $\downarrow 14.4$	15.9	1.5 $\downarrow 14.4$	38.1	8.6 $\downarrow 29.6$	40.2	8.9 $\downarrow 31.3$

Table 1: Controlled prior-conflict diagnostics by task family on Qwen3-8B. Scores are percentages. Green upward annotations denote accuracy gains, and red downward annotations denote reductions in prior-answer hit (PA-hit) and prior-step hit (PS-hit) relative to the base model. Validation uses disjoint in-distribution instances; controlled OOD shifts templates, rule instantiations, entity domains, evidence layouts, context structures, and shortcut patterns. Qwen2.5-7B-1M diagnostics are in Appendix D.1.

trolled OOD accuracy improves from 36.0% to 65.4%, while OOD PA-hit and PS-hit decrease from 18.9%/43.1% to 3.7%/13.3%. Full Qwen2.5-7B-1M family-level results are provided in Appendix D.1.

Downstream transfer and attribution. Table 2 reports transfer to CL-Bench, RULER-QA, and LongBench v2. For RL methods, we report mean $\pm$ standard deviation over three independent seeds. On Qwen3-8B, outcome-only RL already improves over the base model, confirming that generic RL contributes to the transfer gains. Adding process-level supervision improves further, while Full Anti-Prior RL remains strongest on all three benchmarks. The same multi-seed advantage over LongRLVR holds for Qwen2.5-7B-1M. We therefore use these benchmarks as evidence of *transfer beyond the synthetic conflict generator*, not as direct measurements of prior suppression. Appendices D.2, D.3, and D.4 provide category- and length-level breakdowns.

Comparison with SFT and LongRLVR. The gains over SFT indicate that merely exposing the

model to anti-prior examples is not sufficient. This is most visible on CL-Bench for Qwen3-8B, where SFT changes the overall score only slightly (6.16 to 6.27). We interpret this as a limitation of likelihood training for the present failure mode: SFT can teach the format and surface traces of anti-prior examples, but it does not directly compare sampled context-grounded trajectories against prior-consistent alternatives, nor does it penalize shortcut steps that remain likely under the base model. The SFT baseline uses the same training prompts and comparable adapter settings, so the weak CL-Bench gain is not used as evidence that SFT is poorly tuned; rather, it motivates the need for trajectory-level reward signals. Section 4.3 further controls for the sampled-completion budget with Reward-weighted SFT, which reuses the full RL rollout log as a same-rollout offline distillation baseline.

The gains over LongRLVR and the downstream RL ablations further suggest that generic optimization and evidence grounding do not fully explain transfer. Outcome-only RL improves performance, and process supervision contributes additional gains, but Full Anti-Prior RL remains

Model	Method	CL-Bench	RULER-QA	LongBench v2
Qwen3-8B	Base	6.16	83.8	36.2
	SFT	6.27	85.3	37.0
	Outcome-only RL	6.67 $\pm$ 0.16	87.13 $\pm$ 0.51	38.10 $\pm$ 0.36
	+ Process reward	8.57 $\pm$ 0.26	91.30 $\pm$ 0.36	41.10 $\pm$ 0.46
	LongRLVR	7.56 $\pm$ 0.37	90.63 $\pm$ 0.40	39.37 $\pm$ 0.45
	Anti-Prior RL	10.58$\pm$0.48	91.90$\pm$0.53	43.07$\pm$0.59
Qwen2.5-7B-1M	Base	4.37	64.8	33.2
	SFT	4.53	68.3	33.8
	LongRLVR	4.98 $\pm$ 0.21	76.53 $\pm$ 0.45	35.30 $\pm$ 0.46
	Anti-Prior RL	6.76$\pm$0.35	80.37$\pm$0.51	38.87$\pm$0.61

Table 2: Downstream transfer and attribution. Scores are percentages; higher is better. Base and SFT are fixed checkpoints; RL rows report mean $\pm$ standard deviation over three independent seeds. The Qwen3-8B outcome-only and process-reward variants separate generic RL and trajectory-supervision effects from the additional gain of explicit prior suppression.

Variant	Val Acc. $\uparrow$	OOD Acc. $\uparrow$	Val PA-hit $\downarrow$	OOD PA-hit $\downarrow$	Val PS-hit $\downarrow$	OOD PS-hit $\downarrow$
Base	51.7	46.4	16.1	15.9	38.1	40.2
SFT	53.9 $\pm$ 0.1	49.1 $\pm$ 0.1	14.1 $\pm$ 0.1	14.8 $\pm$ 0.1	35.9 $\pm$ 0.1	38.7 $\pm$ 0.2
Reward-weighted SFT	68.4 $\pm$ 0.1	63.7 $\pm$ 0.1	7.7 $\pm$ 0.1	7.1 $\pm$ 0.1	19.5 $\pm$ 0.2	21.4 $\pm$ 0.2
Outcome-only RL	58.2 $\pm$ 0.3	54.3 $\pm$ 0.8	11.9 $\pm$ 0.1	11.6 $\pm$ 0.3	28.4 $\pm$ 0.4	30.9 $\pm$ 0.9
+ Context anchoring	61.1 $\pm$ 0.2	57.4 $\pm$ 0.3	10.8 $\pm$ 0.1	10.6 $\pm$ 0.2	24.7 $\pm$ 0.2	26.1 $\pm$ 0.6
+ Process reward	69.8 $\pm$ 0.4	66.9 $\pm$ 0.6	7.2 $\pm$ 0.1	6.9 $\pm$ 0.3	18.2 $\pm$ 0.3	19.3 $\pm$ 0.7
Full Anti-Prior RL	74.5$\pm$0.5	73.4$\pm$0.8	1.6$\pm$0.1	1.5$\pm$0.2	8.6$\pm$0.2	8.9$\pm$0.6

Table 3: Grouped reward ablation and Reward-weighted SFT control on Qwen3-8B controlled validation and controlled OOD splits. Scores are percentages; trainable variants report mean $\pm$ standard deviation over three seeds. Reward-weighted SFT reuses the full rollout log collected during Anti-Prior RL and trains from the SFT checkpoint with a reward-weighted supervised objective. The RL variants form a nested hierarchy: outcome reward, context anchoring, process reward, and prior-answer/prior-step penalties.

strongest. This pattern is consistent with an additional benefit from explicitly identifying and suppressing prior-dominated trajectories. Because the downstream benchmarks are not designed specifically for context-prior conflict, we interpret these results only as evidence that the learned policy shift transfers beyond the constructed generator.

4.3 Grouped Reward Ablation

Table 3 isolates the reward components under matched training settings and includes Reward-weighted SFT as a same-rollout offline control. SFT yields only a modest OOD gain (49.1%), whereas Reward-weighted SFT reaches 63.7%, showing that high-reward trajectories are partially distillable. Process reward improves OOD accuracy further to 66.9%, and Full Anti-Prior RL reaches 73.4% while reducing OOD PS-hit to 8.9%. Thus, access to the same sampled-completion pool is not sufficient: the remaining gain is associated with on-policy optimization together with process supervision and explicit shortcut penalties.

4.4 Controlled OOD and Family-Level Analysis

The controlled gains are not concentrated in a single synthetic pattern. Anti-Prior RL improves every Qwen3-8B family on validation and OOD while reducing PA-hit and PS-hit. The largest OOD accuracy gains occur on structured-table (+46.0), ordering/timeline (+37.5), and meeting/query (+37.0), which also exhibit large PS-hit reductions. The OOD split remains an *in-family* shift rather than a held-out-task evaluation: it changes templates, rule instantiations, entity domains, evidence layouts, context structures, and shortcut patterns while preserving the high-level task family. We therefore interpret these verifier-defined diagnostics jointly with independent validation below, rather than as objective-independent evidence by themselves.

4.5 Independent Validation and Instance-Level Attribution

Because PA-hit and PS-hit are operationally related to the reward, we perform three complementary analyses that do not rely solely on the training-time verifier.

Independent prior-use judge. On the controlled OOD split, an independent LLM judge (DeepSeek-V4-Pro) receives neither reward definitions nor verifier outputs and directly labels whether the final answer (*PA-use*) or intermediate reasoning (*PS-use*) follows the prior shortcut. Table 4 shows close agreement with the verifier (overall agreement $> 93\%$) and the same reduction trend: Full Anti-Prior RL lowers judge PA-use from 16.6% to 1.6% and judge PS-use from 43.8% to 10.4%. This makes it unlikely that the headline reduction is solely a consequence of optimizing the original detector or exploiting its contrast exemption. The semantic agreement is especially useful for PS-use, where the independent judge can recognize prior reliance even when the wording differs from the verifier’s stored shortcut representation.

Model	V-PA	J-PA	V-PS	J-PS
Base	15.9	16.6	40.2	43.8
Outcome RL	11.6	11.9	30.9	34.1
Full Anti-Prior RL	1.5	1.6	8.9	10.4

Table 4: Independent prior-use validation on controlled OOD. V/J denote verifier/judge rates.

Instance-level attribution. Table 5 reports the instance-level correction analysis. Among base-model OOD errors with PS-hit, Full Anti-Prior RL corrects 372/557 (66.8%), compared with only 6/193 (3.1%) errors without PS-hit. Moreover, 372 of the 438 removed PS-hit instances become correct (84.9%). Thus, the accuracy improvement is

concentrated on the prior-dominated failures targeted by the method rather than uniformly across unrelated errors.

Base-model failure type	N	Corrected	Rate
Error $\cap$ PS-hit	557	372	66.8
Error $\cap \neg$ PS-hit	193	6	3.1

Table 5: Base-error correction by Full Anti-Prior RL, partitioned by PS-hit.

No-conflict control. Finally, we remove the overriding contextual rule while retaining task structure. In this neutral setting, the base model recovers the constructed prior answer and reasoning path at **81.9%** and **94.1%**, respectively. This supports the interpretation that the generator’s shortcut paths track genuine default model preferences rather than arbitrary planted alternatives.

Triangulating the mechanism. These checks address complementary alternatives: the independent judge tests whether prior-use reductions survive an unseen semantic evaluator; instance-level conditioning shows that corrections concentrate on prior-dominated errors; and the no-conflict control verifies that the constructed shortcut is attractive to the base model when the contextual override is absent. None alone identifies a latent causal prior, but together they substantially narrow detector-optimization and arbitrary-shortcut explanations. Additional parser-robustness and reward-sensitivity checks are reported in Appendix C.4.

4.6 Interpreting the Evidence

The added analyses separate three explanations that aggregate accuracy alone cannot distinguish. First, the improvement is not simply a consequence of exposing the model to more sampled completions: Reward-weighted SFT can distill part of the roll-out pool, but remains below the process-reward RL variant and Full Anti-Prior RL on controlled OOD. Second, the reduction in prior-hit is not supported only by the training-time verifier. The independent judge reproduces the same PA/PS trend, while the no-conflict control shows that the stored shortcut paths are highly attractive to the base model when the contextual override is removed. Third, the instance-level analysis links these two observations: corrections are concentrated on errors that exhibit prior-step reliance, rather than being spread uniformly across all base-model failures.

These results also clarify the role of the downstream benchmarks. CL-Bench, RULER-QA, and LongBench v2 do not explicitly instantiate our synthetic context-prior conflicts, so their gains should not be read as direct measurements of PA/PS suppression. Instead, they test whether the learned policy shift transfers beyond the generator to independently constructed tasks requiring context learning, long-context question answering, and realistic long-context reasoning. The consistent improvements across two Qwen-family models, together with the controlled OOD shifts and independent prior-use diagnostics, therefore support a narrower conclusion: when a conflicting shortcut can be specified and verified during training, suppressing that shortcut can improve context-grounded reasoning without confining the gains to the synthetic templates used for optimization.

5 Conclusion

We study prior-dominated reasoning, where models follow prior-consistent answers or steps even when local context contradicts them. Anti-Prior Reinforcement Learning provides a verifier-based framework for observable and constructible conflict settings, using answer correctness, context anchoring, step-wise rule use, and explicit shortcut suppression. Controlled evaluation, independent prior-use judging, instance-level attribution, and no-conflict controls jointly support the intended mechanism. Multi-seed downstream results further show transfer beyond the synthetic generator, although those benchmarks do not directly measure context-prior conflict. These findings suggest that explicitly modeling and suppressing verifiable shortcut trajectories can improve context-grounded reasoning when the relevant conflict structure can be specified.

Limitations

Our work has several limitations. First, Anti-Prior RL requires an observable, constructible conflict structure and a verifier schema; it is therefore a diagnosis and training framework for verifiable conflict settings rather than a universal mitigation method for arbitrary real-world datasets. The synthetic task families do not cover all model priors, and an unseen conflict family with a newly constructed generator/verifier remains future work. Second, our verifier provides partial but targeted supervision: it can identify known gold paths and

high-risk prior paths, but not every possible incorrect rationale, and valid alternative derivations outside the verifier schema may receive only partial process credit. Third, experiments are limited to two Qwen-family models, and the independent prior-use analysis relies on a model-based judge rather than human annotation; broader model-family and human validation would strengthen the evidence. Fourth, stronger context following can interact with safety alignment. Our objective targets task-level semantic, procedural, and reasoning shortcuts rather than safety policies, but this distinction is induced by the training data and verifier scope rather than formally guaranteed by the architecture. Finally, downstream benchmark gains show transfer under the evaluated settings but do not directly measure context-prior conflict or establish universal causal understanding of context use.

Ethical Considerations

This work aims to improve LLMs’ ability to follow task-relevant context rather than rely on misleading reasoning shortcuts. Stronger context following is nevertheless dual-use: a model may also become more responsive to harmful or deceptive contextual instructions if higher-priority safety policies are not preserved. Anti-Prior RL does not explicitly reward overriding safety policies; its verifier is restricted to controlled task-level rules and evidence. We therefore treat preservation of higher-priority safety behavior as outside the formal guarantees of the present method. A preliminary safety check is reported in Appendix C.4; broader safety evaluation is needed before deployment.

Our synthetic training data are generated from templates and use synthetic entities, rules, and contexts rather than human-authored personal text. We apply generation-time filters and manual audits to check that generated instances do not contain real personally identifying information or offensive content; since the data are template-generated and do not include real user records, no additional anonymization is required. We provide supplementary materials containing the core verifier/reward implementation, controlled validation and OOD evaluation sets, schema files, and documentation. We use third-party model, benchmark, and baseline artifacts only for research evaluation and comparison, consistent with their original public releases, licenses, and terms of use. Artifacts created

in this work are intended for research on context-grounded reasoning and controlled evaluation of context-prior conflicts, not for direct deployment or for evaluating individuals. We plan to release code, data generation scripts, controlled evaluation sets, and evaluation tools with accompanying documentation, license, copyright notice, and terms of use where permitted.

Acknowledgments

AI assistants were used for language polishing, manuscript editing, and limited coding assistance. All research decisions, experimental design, analyses, experimental results, and scientific claims were made or verified by the authors. All AI-assisted content was reviewed and edited by the authors, who take full responsibility for the final manuscript.

References

- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, and 32 others. 2022. [Constitutional AI: Harmlessness from AI feedback](#). *arXiv preprint arXiv:2212.08073*.
- Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. 2024. [LongAlign: A recipe for long context alignment of large language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1376–1395.
- Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2025. [LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664.
- Amanda Bertsch, Maor Ivgi, Emily Xiao, Uri Alon, Jonathan Berant, Matthew R. Gormley, and Graham Neubig. 2025. [In-context learning with long-context models: An in-depth exploration](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 12119–12149.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu,

- Clemens Winter, and 12 others. 2020. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.
- Guanzheng Chen, Michael Qizhe Shieh, and Lidong Bing. 2026. [LongRLVR: Long-context reinforcement learning requires verifiable context rewards](#). In *The Fourteenth International Conference on Learning Representations*.
- Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. [Deep reinforcement learning from human preferences](#). In *Advances in Neural Information Processing Systems*, volume 30.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Łukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *arXiv preprint arXiv:2110.14168*.
- DeepSeek-AI. 2025. [DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning](#). *arXiv preprint arXiv:2501.12948*.
- Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, KaShun Shum, and Tong Zhang. 2023. [RAFT: Reward ranked finetuning for generative foundation model alignment](#). *Transactions on Machine Learning Research*.
- Shihan Dou, Ming Zhang, Zhangyue Yin, Chenhao Huang, Yujiong Shen, Junzhe Wang, Jiayi Chen, Yuchen Ni, Junjie Ye, Cheng Zhang, Huaibing Xie, Jianglu Hu, Shaolei Wang, Weichao Wang, Yanling Xiao, Yiting Liu, Zenan Xu, Zhen Guo, Pluto Zhou, and 8 others. 2026. [CL-Bench: A benchmark for context learning](#). *arXiv preprint arXiv:2602.03587*.
- Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. 2025. [How to train long-context language models \(effectively\)](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7376–7399.
- Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. 2020. [Shortcut learning in deep neural networks](#). *Nature Machine Intelligence*, 2(11):665–673.
- Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. 2023. [Reinforced self-training \(ReST\) for language modeling](#). *arXiv preprint arXiv:2308.08998*.
- Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. [Annotation artifacts in natural language inference data](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112.
- Cheng-Ping Hsieh, Simeng Sun, Samuel Krizan, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. [RULER: What’s the real context size of your long-context language models?](#) In *Proceedings of COLM 2024*.
- Robin Jia and Percy Liang. 2017. [Adversarial examples for evaluating reading comprehension systems](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.
- Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošūūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, and 11 others. 2023. [Measuring faithfulness in chain-of-thought reasoning](#). *arXiv preprint arXiv:2307.13702*.
- Yanyang Li, Shuo Liang, Michael R. Lyu, and Liwei Wang. 2024. [Making long-context language models better multi-hop reasoners](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2462–2475.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *International Conference on Learning Representations*.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. [Lost in the middle: How language models use long contexts](#). *Transactions of the Association for Computational Linguistics*, 12:157–173.
- Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. [Entity-based knowledge conflicts in question answering](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7052–7063.
- R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. [Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448.
- Sewon Min, Xinxin Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. [Rethinking the role of demonstrations: What makes in-context learning work?](#) In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064.

- Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. [Show your work: Scratchpads for intermediate computation with language models](#). *arXiv preprint arXiv:2112.00114*.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#). In *Advances in Neural Information Processing Systems*.
- Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. 2019. [Advantage-weighted regression: Simple and scalable off-policy reinforcement learning](#). *arXiv preprint arXiv:1910.00177*.
- Jan Peters and Stefan Schaal. 2007. [Reinforcement learning by reward-weighted regression for operational space control](#). In *Proceedings of the 24th International Conference on Machine Learning*, pages 745–750.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. [Direct preference optimization: Your language model is secretly a reward model](#). In *Advances in Neural Information Processing Systems*.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. [Proximal policy optimization algorithms](#). *arXiv preprint arXiv:1707.06347*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [DeepSeekMath: Pushing the limits of mathematical reasoning in open language models](#). *arXiv preprint arXiv:2402.03300*.
- Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. [Large language models can be easily distracted by irrelevant context](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.
- Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. 2020. [Learning to summarize from human feedback](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021.
- Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Y. Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. [Solving math word problems with process- and outcome-based feedback](#). *arXiv preprint arXiv:2211.14275*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. [Self-consistency improves chain of thought reasoning in language models](#). In *International Conference on Learning Representations*.
- Yihan Wang, Andrew Bai, Nanyun Peng, and Cho-Jui Hsieh. 2025. [On the loss of context awareness in general instruction fine-tuning](#). In *Advances in Neural Information Processing Systems*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*.
- Xiaodong Wu, Minhao Wang, Yichen Liu, Xiaoming Shi, He Yan, Lu Xiangju, Junmin Zhu, and Wei Zhang. 2025. [LIFBench: Evaluating the instruction following performance and stability of large language models in long-context scenarios](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16445–16468.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. [Qwen3 technical report](#). *arXiv preprint arXiv:2505.09388*.
- An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, and 9 others. 2025b. [Qwen2.5-1M technical report](#). *arXiv preprint arXiv:2501.15383*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. [ReAct: Synergizing reasoning and acting in language models](#). In *International Conference on Learning Representations*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. [DAPO: An open-source LLM reinforcement learning system at scale](#). *arXiv preprint arXiv:2503.14476*.
- Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 2023. [RRHF: Rank responses to align language models with human feedback](#). In *Advances in Neural Information Processing Systems*.

- Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. [STaR: Bootstrapping reasoning with reasoning](#). In *Advances in Neural Information Processing Systems*.
- Jiajie Zhang, Zhongni Hou, Xin Lv, Shulin Cao, Zhenyu Hou, Yilin Niu, Lei Hou, Yuxiao Dong, Ling Feng, and Juanzi Li. 2025. [LongReward: Improving long-context large language models with AI feedback](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3718–3739.
- Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. [Calibrate before use: Improving few-shot performance of language models](#). In *Proceedings of the 38th International Conference on Machine Learning*, pages 12697–12706.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. [Least-to-most prompting enables complex reasoning in large language models](#). In *International Conference on Learning Representations*.
- Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul F. Christiano, and Geoffrey Irving. 2019. [Fine-tuning language models from human preferences](#). *arXiv preprint arXiv:1909.08593*.
- Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. [Universal and transferable adversarial attacks on aligned language models](#). *Preprint*, arXiv:2307.15043.

A Additional Details on Data Generation

Latent program fields. Each anti-prior instance is generated by an executable program that stores both context-conditioned and prior-consistent supervision. The policy observes only the rendered prompt. The hidden schema contains `gold_answer`, `prior_shortcut_answers`, `expected_intermediate_steps`, `prior_intermediate_steps`, and `rule/evidence markers`. These fields provide verifier-only supervision for answer checking, process matching, prior-hit diagnostics, and data validation.

Executable conflict construction. For each family, the generator samples entities, contextual rules, evidence blocks, query variables, and a conflict configuration, then runs two deterministic solvers over the same latent specification. The *context solver* produces the gold answer and trajectory from the local rule or evidence, whereas the *shortcut solver* applies the plausible default assumption. We retain an instance only when the two solvers disagree. Consequently, the gold and prior trajectories are paired outputs of one executable task specification rather than independently written rationales.

Dataset characterization. The seven task families target distinct shortcut mechanisms rather than merely changing surface templates. Each family contains 3,000 training, 200 validation, and 200 controlled-OOD examples. Validation uses disjoint in-distribution instances, whereas controlled OOD preserves the high-level capability while shifting templates, rule instantiations, entity domains, evidence layouts, context structures, and shortcut patterns. Table 6 summarizes the targeted shortcut type, split size, and verifier-relevant structural statistics for every family.

Filtering and quality control. We require the gold path to reproduce the stored answer, the prior path to disagree with it, and all required rule/evidence markers to occur in the rendered prompt. Ambiguous instances are discarded.

Task family	Targeted prior shortcut	Train/Val/OOD	Avg. markers	Avg. gold steps
Single-doc scoped QA	Salient but out-of-scope evidence	3000/200/200	9.0	4.0
Meeting/query summary	Query-agnostic/global summary	3000/200/200	9.0	4.0
Code/config dependency	Default dependency/config resolution	3000/200/200	9.0	4.0
Few-shot rule transfer	Memorized surface mapping pattern	3000/200/200	9.0	4.0
Passage count unique	Counting repeated mentions instead of unique entities	3000/200/200	6.0	3.0
Ordering/timeline query	Surface order instead of explicit constraints	3000/200/200	12.2	4.0
Structured table conditional	Default/majority table-reading heuristic	3000/200/200	6.0	5.0

Table 6: Characterization of the generated anti-prior data. Each task family targets a distinct prior-consistent shortcut. Markers are task-program-defined rule/evidence markers used for verifier grounding.

Canonical gold trajectories must also score above prior trajectories under the verifier. Finally, we manually audit samples from every family for prompt clarity, label correctness, and the plausibility of the intended shortcut.

B Theoretical Perspective

This appendix provides an idealized perspective on why the proposed reward can reduce prior-dominated reasoning. The analysis is not intended as a completeness proof for the verifier, nor as a guarantee that arbitrary real-world priors can be removed. Instead, it formalizes the main design intuition under simplified assumptions: context anchoring and process rewards provide local credit to context-grounded trajectories, while prior-answer and prior-step penalties explicitly move probability mass away from prior-consistent shortcut trajectories.

B.1 Formalization

Let $X = (Q, C)$ denote a query-context pair, where Q is the query and C is the current context. A model completion is written as

$$Y = (T, A),$$

where T is the reasoning trajectory and A is the final answer. For each controlled anti-prior instance, the task program defines a context-conditioned gold mechanism (T^*, A^*) and one or more prior-consistent shortcut mechanisms $(T^{\text{pr}}, A^{\text{pr}})$. The verifier gives positive credit to trajectories that answer correctly, identify the relevant contextual source, and apply the local rule, while penalizing completions whose final answer or intermediate steps match the prior-consistent path.

For compact notation, let

$$\begin{aligned} r_a &= r_{\text{ans}}(x, y), & r_c &= r_{\text{anchor}}(x, y), \\ r_p &= r_{\text{proc}}(x, y), & h_a &= h_a(x, y), \\ & & h_s &= h_s(x, y). \end{aligned}$$

The positive context-grounded utility and shortcut penalty are

$$\begin{aligned} U_x(y) &= g(y)(\lambda_a r_a + \lambda_c r_c + \lambda_p r_p), \\ V_x(y) &= \lambda_{pa} h_a + \lambda_{ps} h_s. \end{aligned}$$

Ignoring clipping only for notational clarity, the reward is

$$R(x, y) = U_x(y) - V_x(y).$$

The real implementation additionally uses clipping and answer-conditioned caps; these do not change the qualitative role of the terms, but make the analysis less transparent.

We analyze a KL-regularized contextual policy objective for a fixed input x :

$$\begin{aligned} J_x(\pi) &= \sum_{y \in \mathcal{Y}(x)} \pi(y | x) (U_x(y) - V_x(y)) \\ &\quad - \tau \text{KL}(\pi(\cdot | x) \| \pi_{\text{ref}}(\cdot | x)), \end{aligned}$$

where $\tau > 0$ controls the strength of reference-policy regularization. This is an idealized one-step view of the sequence-level verifier reward used in GRPO-style training.

B.2 Prior Penalties Suppress Shortcut Mass

Proposition 1. For a fixed input x , the optimizer of $J_x(\pi)$ satisfies

$$\pi^*(y | x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y | x) \exp\left(\frac{U_x(y) - V_x(y)}{\tau}\right),$$

where $Z(x)$ is a normalizing constant. Moreover, let

$$\begin{aligned} \mathcal{S}_\Delta(x) &= \{y : V_x(y) \geq \Delta\}, \\ \mathcal{G}_0(x) &= \{y : V_x(y) = 0\}, \end{aligned}$$

where $\Delta > 0$. Then

$$\begin{aligned} \frac{\pi^*(\mathcal{S}_\Delta(x) | x)}{\pi^*(\mathcal{G}_0(x) | x)} &\leq \exp(-\Delta/\tau) \frac{N_\Delta(x)}{D_0(x)}, \\ N_\Delta(x) &= \sum_{y \in \mathcal{S}_\Delta(x)} \pi_{\text{ref}}(y | x) \exp(U_x(y)/\tau), \\ D_0(x) &= \sum_{y \in \mathcal{G}_0(x)} \pi_{\text{ref}}(y | x) \exp(U_x(y)/\tau). \end{aligned}$$

Derivation. Introducing a Lagrange multiplier for $\sum_y \pi(y | x) = 1$ gives

$$\begin{aligned} \mathcal{L} &= \sum_y \pi(y) (U_x(y) - V_x(y)) \\ &\quad - \tau \sum_y \pi(y) \log \frac{\pi(y)}{\pi_{\text{ref}}(y)} \\ &\quad + \lambda \left(\sum_y \pi(y) - 1 \right). \end{aligned}$$

Setting the derivative with respect to $\pi(y)$ to zero yields

$$\pi^*(y | x) \propto \pi_{\text{ref}}(y | x) \exp\left(\frac{U_x(y) - V_x(y)}{\tau}\right).$$

For $y \in \mathcal{S}_\Delta(x)$, $V_x(y) \geq \Delta$, so

$$\begin{aligned} \exp\left(\frac{U_x(y) - V_x(y)}{\tau}\right) \\ \leq \exp(-\Delta/\tau) \exp(U_x(y)/\tau). \end{aligned}$$

For $y \in \mathcal{G}_0(x)$, $V_x(y) = 0$. Substituting these two expressions into the ratio of shortcut mass to non-penalized mass gives the stated bound.

This proposition formalizes the role of the prior penalties. They are not merely additional reward terms; under a KL-regularized policy update, they reweight the completion distribution against prior-consistent modes. The suppression is exponential in the penalty-to-temperature ratio Δ/τ , which matches the empirical role of the prior-answer and prior-step penalties in reducing PA-hit and PS-hit.

B.3 Process Rewards Improve Credit Assignment

Outcome-only reward can be weak for learning early context-grounding decisions in multi-step reasoning. To illustrate this, consider a stylized H -step prior-conflict task. At the first step, the model chooses a grounding source g , where only g^* is context-correct. At later steps, it applies local rules.

Let

$$\begin{aligned} G &= \{g = g^*\}, \quad E_t = \{s_{2:t} = s_{2:t}^*\}, \\ \Pi_{i:j} &= \prod_{k=i}^j \rho_k, \end{aligned}$$

where

$$\begin{aligned} \alpha &= \pi_\theta(g^* | x), \\ \rho_t &= \Pr_\theta(s_t = s_t^* | G, E_{t-1}, x). \end{aligned}$$

Let z be the logit for choosing g^* , so that $\partial\alpha/\partial z = \alpha(1 - \alpha)$.

With outcome-only reward,

$$R_{\text{out}} = \lambda_a \mathbf{1}\{G, E_H\},$$

the expected reward contribution is

$$J_{\text{out}} = \lambda_a \alpha \Pi_{2:H},$$

and therefore

$$\frac{\partial J_{\text{out}}}{\partial z} = \lambda_a \alpha (1 - \alpha) \Pi_{2:H}.$$

If each downstream step succeeds with probability at most $\bar{\rho} < 1$, this signal decays as $O(\bar{\rho}^{H-1})$.

Now add context anchoring and process rewards:

$$R_{\text{AP}} = \lambda_c \mathbf{1}\{G\} + \sum_{t=2}^H \lambda_p \mathbf{1}\{G, E_t\} + R_{\text{out}}.$$

Then the gradient for the initial grounding decision is

$$\frac{\partial J_{\text{AP}}}{\partial z} = \alpha (1 - \alpha) \left[\lambda_c + \sum_{t=2}^H \lambda_p \Pi_{2:t} + \lambda_a \Pi_{2:H} \right].$$

In particular,

$$\frac{\partial J_{\text{AP}}}{\partial z} \geq \alpha (1 - \alpha) \lambda_c.$$

Thus, anchoring and process rewards provide local learning signals even when the complete reasoning chain has not yet succeeded. This explains why outcome-only RL can improve final accuracy but still leave high prior-step hit rates, while adding process supervision more directly encourages the desired context-conditioned trajectory.

B.4 Information and Causal Interpretation

The same reward decomposition also has an information-theoretic interpretation. Let P denote a task-program-defined descriptor of the prior shortcut. Consider an idealized objective with a context probe $q_c(Y \mid C, Q)$ and a prior probe $q_p(Y \mid P, Q)$:

$$\mathcal{J}_{\text{IT}}(\pi) = \alpha \sup_{q_c} \mathbb{E}_{\pi} [\log q_c(Y \mid C, Q)] + \beta \inf_{q_p} \mathbb{E}_{\pi} [-\log q_p(Y \mid P, Q)].$$

If the probe families are expressive enough to match the induced conditional distributions, then

$$\mathcal{J}_{\text{IT}}(\pi) = \alpha I_{\pi}(Y; C \mid Q) - \beta I_{\pi}(Y; P \mid Q) + (\beta - \alpha) H_{\pi}(Y \mid Q).$$

When $\alpha = \beta$, this reduces to

$$\mathcal{J}_{\text{IT}}(\pi) = \alpha (I_{\pi}(Y; C \mid Q) - I_{\pi}(Y; P \mid Q)).$$

This is not the literal training objective, but it clarifies the direction of the verifier design: anchoring and process rewards encourage completions to carry more information about the current context, while prior-answer and prior-step penalties reduce dependence on shortcut descriptors.

A complementary causal view is to distinguish the desired path

$$C \rightarrow T \rightarrow A$$

from the shortcut path

$$P \rightarrow T \rightarrow A.$$

Assume that, on prior-conflict examples, there exists a deterministic gold process $T^* = t^*(C, Q, U)$ such that when $T = T^*$, the answer is determined by (C, Q, U) and is independent of P . If

$$\Pr(T = T^* \mid X \in \mathcal{X}_{\text{cf}}) \geq 1 - \varepsilon,$$

then for any two prior interventions p, p' ,

$$\begin{aligned} \text{TV}(\mathcal{L}(A \mid \text{do}(P = p), C, Q, U), \\ \mathcal{L}(A \mid \text{do}(P = p'), C, Q, U)) \leq 2\varepsilon. \end{aligned}$$

On the event $T = T^*$, the final answer is a function of the current context and latent task state rather than the prior shortcut. The two interventions can differ only on the failure event $T \neq T^*$, whose probability is at most ε under each intervention. Therefore, reducing process errors and prior-step hits makes the answer less sensitive to shortcut interventions.

This perspective also clarifies the role of our controlled OOD split. The OOD split preserves the high-level context-conditioned mechanism while shifting surface templates, rule instantiations, evidence layouts, and shortcut patterns. If training reduces reliance on the $P \rightarrow T \rightarrow A$ path and increases reliance on the $C \rightarrow T \rightarrow A$ path, the policy should be more robust under these controlled shifts.

B.5 Scope of the Analysis

The analysis above relies on idealized assumptions: the verifier must identify a useful subset of gold and prior trajectories, the controlled OOD shift should preserve the gold mechanism, and KL-regularized policy optimization is only an abstraction of the implemented GRPO-style updates. We therefore use this section as a theoretical perspective rather

than a formal guarantee of universal reasoning correctness. Under these assumptions, however, the reward design has a clear mechanism: positive context-grounded terms increase the value of trajectories that use the current context, process rewards provide denser credit assignment, and prior penalties exponentially suppress known shortcut modes.

C Reward and Training Details

C.1 Reward Implementation Details

The verifier is intentionally partial. It checks whether a completion follows known gold context-conditioned paths and whether it activates known high-risk prior paths; it does not attempt to prove arbitrary reasoning traces correct. A completion that reaches the correct answer through an unrecognized derivation may receive answer and anchoring credit but not full process credit. This makes the reward targeted and auditable rather than a general proof checker.

The verifier-compatibility gate $g(y) \in \{0, 1\}$ is a binary parser-compatibility gate. It equals 1 when the completion contains a parseable final answer and a verifier-compatible reasoning field, and 0 otherwise. When $g(y) = 0$, the positive reward is suppressed; prior-answer and prior-step penalties are still applied when they are detectable from the completion.

The process reward stops accumulating after the first context-inconsistent or prior-driven step. The answer-conditioned cap $C(x, y)$ prevents high anchoring or process scores from compensating for fundamental semantic failures. If the final answer is incorrect, the positive reward is capped at 0.30; if the output matches a known prior-driven answer, the positive reward is capped at 0.10 before prior penalties are applied. We penalize prior shortcuts only when they are used as active inference steps, not when the model explicitly contrasts them with the local contextual rule.

We use fixed reward weights across all task families: $\lambda_a = 0.25$, $\lambda_c = 0.20$, $\lambda_p = 0.35$, $\lambda_{pa} = 0.25$, and $\lambda_{ps} = 0.35$. These weights are fixed before downstream evaluation and are not tuned on CL-Bench, RULER-QA, or LongBench v2. We do not use task-family-specific reward weights.

Practical component computation. The verifier parses each completion into a final answer, contextual rule/evidence usage, and intermediate reasoning states. Matching uses type-aware normaliza-

tion over strings, numbers, booleans, and list/set answers rather than raw keyword overlap. Table 7 summarizes the implementation.

Component	Computation
R_{ans}	Normalized final answer matches gold_answer.
R_{anchor}	Coverage of required contextual rule/evidence markers.
R_{proc}	Agreement with normalized gold intermediate states; credit stops after the first inconsistent/prior-driven step.
P_{pa}	Final answer matches a stored prior-shortcut answer.
P_{ps}	Generated intermediate states match active prior-shortcut states.

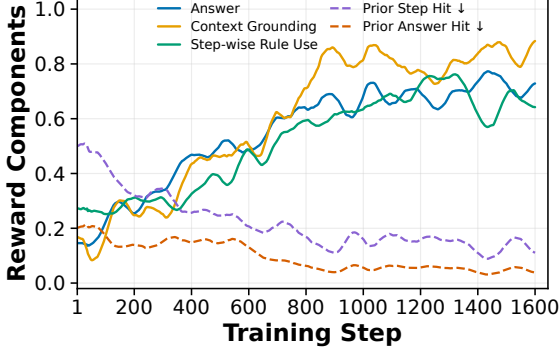
Table 7: Practical verifier computation.

C.2 Optimization and Inference Details

We use LoRA adapters with rank $r = 16$, $\alpha = 32$, and dropout 0.05. Unless otherwise specified, Anti-Prior RL is trained for 1600 GRPO update steps with 8 generations per prompt, learning rate 5×10^{-6} , KL coefficient $\beta = 0.001$, per-device train batch size 1, gradient accumulation steps 8, and 2 NVIDIA A800 GPUs (80GB each). A complete Anti-Prior RL run takes approximately 36 hours of wall-clock time, corresponding to about 72 GPU-hours. The run contains 25.6K prompt presentations and 204.8K sampled completions, or approximately 1.22 prompt-level epochs over the 21K training prompts.

For Reward-weighted SFT, we reuse the complete rollout log collected during Anti-Prior RL. The model is initialized from the SFT checkpoint, verifier-compatible completions are weighted by their verifier rewards, and training uses a supervised objective rather than GRPO. This baseline is a same-rollout offline control: it uses the same sampled completions and verifier scores as RL, but it does not regenerate rollouts from the SFT policy and does not perform group-relative policy-gradient updates.

For rollouts, we set the maximum completion length to 2048, temperature to 0.7, top- p to 0.9, and no-repeat n -gram size to 8. The verifier reward is computed at the sequence level and normalized within each rollout group to obtain group-relative advantages. For controlled prior-conflict evaluation, we sample three completions per example with the same temperature and top- p settings and use a strict JSON parser to extract answers and verifier diagnostics. Downstream benchmarks follow the official evaluation protocols where available and use the same decoding and answer extraction rules across compared methods.



(a) Reward components



(b) Total reward

Figure 3: Reward diagnostics during Anti-Prior RL training. Positive reward components increase, prior-hit terms decrease, and the total verifier reward rises steadily, indicating a stable non-degenerate learning signal.

LongRLVR implementation. Our LongRLVR reimplementation uses the same base checkpoints, LoRA configuration, rollout strategy, optimization budget, and downstream evaluation protocol as Anti-Prior RL. The primary difference is the reward: LongRLVR uses verifiable answer/context-utilization signals, whereas Anti-Prior RL additionally represents prior-consistent trajectories and applies explicit prior-answer/prior-step penalties. This matched protocol is intended to isolate the contribution of explicit prior suppression rather than optimization-budget differences.

C.3 Reward Validation and Training Diagnostics

We examine whether the verifier reward behaves as intended during training. This is important because our method relies on a structured reward rather than an answer-only signal: a useful verifier should increase context-grounded components while reducing prior-driven behavior. We therefore track the main reward components over training, including answer correctness, context grounding, step-wise rule use, prior-step hit, and prior-answer hit.

Figure 3 shows that the positive components of the reward increase throughout training: the model becomes more likely to answer correctly, anchor its reasoning in the provided context, and apply local rules step by step. At the same time, both prior-step hit and prior-answer hit decrease. The total verifier reward also rises steadily. This pattern supports the intended mechanism of Anti-Prior RL: the policy is not only improving final-answer accuracy, but also moving away from prior-consistent shortcut trajectories. These curves provide a compact training diagnostic; reward variance, clipping ratios, and

cap activation rates can be tracked with the same logging pipeline.

C.4 Additional Robustness Checks

Lenient answer extraction. Because the controlled evaluator uses a strict parser, we re-evaluate the base model with a lenient format-normalized extractor. Table 8 reports the comparison: validation accuracy changes from 51.7% to 52.3% (0.6% parse failures), and OOD accuracy from 46.4% to 47.2% (0.8% parse failures). Thus, format failures account for only a small fraction of the controlled accuracy gap.

Split	Strict Acc.	Lenient Acc.	Parse fail
Val	51.7	52.3	0.6
OOD	46.4	47.2	0.8

Table 8: Base-model parser robustness. Values are percentages.

Prior-penalty sensitivity. We jointly scale the prior-answer/prior-step penalty coefficients while keeping all other training settings unchanged. OOD accuracy and prior-hit remain stable across $0.5\times$, $1.0\times$, and $2.0\times$ the default penalty strength (Table 9), suggesting that the method does not rely on a narrowly tuned penalty magnitude.

Penalty scale	OOD Acc. $\uparrow$	PA-hit $\downarrow$	PS-hit $\downarrow$
$0.5\times$	71.5	2.1	11.3
$1.0\times$	73.4	1.5	8.9
$2.0\times$	72.8	1.7	9.7

Table 9: Sensitivity to the jointly scaled prior-penalty coefficients on controlled OOD.

Preliminary safety evaluation. To probe whether stronger task-level context following is accompanied by an obvious loss of refusal behavior, we conduct a small-scale evaluation

Task family	Accuracy $\uparrow$				PA-hit $\downarrow$				PS-hit $\downarrow$			
	Val		OOD		Val		OOD		Val		OOD	
	Base	RL	Base	RL	Base	RL	Base	RL	Base	RL	Base	RL
Single-doc scoped QA	42.5	79.5 $\uparrow$ 37.0	41.5	80.0 $\uparrow$ 38.5	18.5	1.0 $\downarrow$ 17.5	21.0	0.5 $\downarrow$ 20.5	41.0	7.5 $\downarrow$ 33.5	45.0	7.0 $\downarrow$ 38.0
Meeting/query summary	35.5	69.5 $\uparrow$ 34.0	36.5	69.0 $\uparrow$ 32.5	23.5	2.5 $\downarrow$ 21.0	21.5	2.0 $\downarrow$ 19.5	53.5	11.0 $\downarrow$ 42.5	49.0	10.0 $\downarrow$ 39.0
Code/config dependency	46.5	78.0 $\uparrow$ 31.5	48.0	79.0 $\uparrow$ 31.0	16.0	0.5 $\downarrow$ 15.5	15.5	1.5 $\downarrow$ 14.0	39.5	5.5 $\downarrow$ 34.0	36.5	6.5 $\downarrow$ 30.0
Few-shot rule transfer	50.0	73.5 $\uparrow$ 23.5	26.0	54.0 $\uparrow$ 28.0	16.5	2.0 $\downarrow$ 14.5	20.0	5.5 $\downarrow$ 14.5	27.5	4.5 $\downarrow$ 23.0	45.5	19.0 $\downarrow$ 26.5
Passage count unique	42.0	59.0 $\uparrow$ 17.0	42.5	58.5 $\uparrow$ 16.0	14.0	5.0 $\downarrow$ 9.0	12.0	3.5 $\downarrow$ 8.5	38.5	15.0 $\downarrow$ 23.5	34.0	12.5 $\downarrow$ 21.5
Ordering/timeline query	46.5	80.0 $\uparrow$ 33.5	45.0	79.5 $\uparrow$ 34.5	16.0	1.5 $\downarrow$ 14.5	17.5	1.5 $\downarrow$ 16.0	36.5	4.0 $\downarrow$ 32.5	37.0	5.0 $\downarrow$ 32.0
Structured table conditional	10.0	47.0 $\uparrow$ 37.0	12.5	37.5 $\uparrow$ 25.0	26.5	9.0 $\downarrow$ 17.5	25.0	11.5 $\downarrow$ 13.5	56.5	24.0 $\downarrow$ 32.5	55.0	33.0 $\downarrow$ 22.0
Overall	39.0	69.5 $\uparrow$ 30.5	36.0	65.4 $\uparrow$ 29.4	18.7	3.1 $\downarrow$ 15.6	18.9	3.7 $\downarrow$ 15.2	41.9	10.2 $\downarrow$ 31.6	43.1	13.3 $\downarrow$ 29.9

Table 10: Qwen2.5-7B-1M controlled diagnostics by task family; scores are percentages.

on 50 harmful instructions sampled from the AdvBench harmful-behavior set (Zou et al., 2023). The base Qwen3-8B model has a jailbreak attack success rate (ASR) of 14%, compared with 16% after Anti-Prior RL. Given the small sample and narrow benchmark scope, this difference is not sufficient to support a strong conclusion about either safety preservation or degradation. We therefore treat this result only as a preliminary check and leave broader evaluation across larger and more diverse jailbreak benchmarks to future work.

D Additional Results

This appendix reports controlled and downstream breakdowns. Bucket/category tables use the default-seed checkpoints for diagnostic detail, whereas Table 2 reports the multi-seed aggregate comparison.

D.1 Controlled Diagnostics on Qwen2.5-7B-1M

The Qwen2.5-7B-1M results mirror Table 1: Anti-Prior RL improves validation and controlled-OOD accuracy while substantially reducing both prior-answer and prior-step hit rates. The largest OOD accuracy gains occur in single-doc scoped QA (+38.5), ordering/timeline (+34.5), meeting/query summary (+32.5), and code/config dependency (+31.0). Table 10 reports the complete family-level breakdown.

D.2 CL-Bench Category-Level Breakdown

CL-Bench contains 1,899 tasks across Domain Knowledge Reasoning, Rule System Application, Procedural Task Execution, and Empirical Discovery & Simulation. Anti-Prior RL improves over LongRLVR in every category for both base models (Table 11), indicating that the aggregate transfer gain is not driven by a single category.

D.3 RULER-QA Length-Bucket Breakdown

The RULER-QA subset contains 200 examples each at 32k, 64k, and 128k. Anti-Prior RL improves over LongRLVR at all three context lengths for both models (Table 12), so the transfer gain is not confined to shorter contexts.

D.4 LongBench v2 Length-Bucket Breakdown

LongBench v2 contains short (180), medium (215), and long (108) context-length buckets. Anti-Prior RL improves over LongRLVR in all three buckets for both models (Table 13), including the longest bucket.

E Qualitative Behavior Examples

The left/right reversal in Figure 1 and a constrained code-generation example illustrate the same mechanism across two very different prior types: a familiar semantic or implementation pattern remains high-probability, but the current context explicitly requires another trajectory.

Case 1: reversed spatial semantics. The local context reverses the meanings of “left” and “right.” The base model can restate this rule correctly yet subsequently derive a conventional physical relation, yielding a context-inconsistent answer. After Anti-Prior RL, the model applies the locally defined reversed semantics consistently across intermediate steps and reaches the context-defined answer. This case shows that the target failure is not simply forgetting the instruction; the model may explicitly mention the instruction while still executing a pretrained semantic shortcut.

Case 2: constrained palindrome implementation. The prompt asks for a Python palindrome checker under three explicit constraints: do not use string slicing [: -1], do not construct a reversed copy, and compare characters manually. The base model tends to fall back to the canonical high-

Model	Method	Domain Knowledge Reasoning	Rule System Application	Procedural Task Execution	Empirical Discovery & Simulation	Overall
Qwen3-8B	Base	6.18	6.89	6.58	3.02	6.16
	SFT	6.33	7.42	6.37	2.51	6.27
	LongRLVR	7.54	8.30	7.64	5.03	7.53
	Anti-Prior RL	10.71	11.31	11.04	7.04	10.58
Qwen2.5-7B-1M	Base	4.52	4.59	4.88	2.01	4.37
	SFT	4.83	4.95	4.67	2.01	4.53
	LongRLVR	5.13	5.48	5.10	2.51	4.95
	Anti-Prior RL	6.94	7.24	7.01	4.02	6.74

Table 11: CL-Bench category-level breakdown. Scores are percentages; higher is better.

Model	Method	32k	64k	128k	Overall
Qwen3-8B	Base	86.0	84.0	81.5	83.8
	SFT	88.0	85.0	83.0	85.3
	LongRLVR	93.0	90.5	88.5	90.7
	Anti-Prior RL	95.0	91.0	89.0	91.7
Qwen2.5-7B-1M	Base	70.0	66.0	58.5	64.8
	SFT	74.5	69.5	61.0	68.3
	LongRLVR	81.5	75.0	73.0	76.5
	Anti-Prior RL	86.5	78.5	76.5	80.5

Table 12: RULER-QA length-bucket breakdown; each bucket contains 200 examples.

Model	Method	Short	Medium	Long	Overall
Qwen3-8B	Base	45.0	30.7	32.4	36.2
	SFT	45.6	29.8	37.0	37.0
	LongRLVR	47.8	32.6	38.9	39.4
	Anti-Prior RL	51.7	36.7	42.6	43.3
Qwen2.5-7B-1M	Base	37.8	31.2	29.6	33.2
	SFT	38.9	31.6	29.6	33.8
	LongRLVR	42.2	32.1	30.6	35.4
	Anti-Prior RL	46.1	34.9	35.2	39.0

Table 13: LongBench v2 length-bucket breakdown (180/215/108 short/medium/long examples).

frequency implementation return `s == s[::-1]`, which is functionally correct but violates the local rule. Anti-Prior RL instead uses a two-pointer comparison from the two ends of the string and therefore satisfies both the functional requirement and the context-defined implementation constraint.

Interpretation. These cases complement the aggregate results because the overridden priors differ qualitatively: one is a conventional semantic relation and the other is a canonical code pattern. In both cases, the goal is not to erase the pretrained prior globally, but to suppress it when an explicit task-relevant contextual rule makes that prior invalid for the current instance.