DIAGNOSING & MITIGATING CONSTRAINT INDUCED RESPONSE COLLAPSE IN INSTRUCTION-TUNED LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Instruction-tuned language models are trained to give thorough, well-organized answers. Recent work showed that this behavior is fragile: banning a single character, such as the em dash, can remove up to 48% of an answer's substantive content. This *constraint-induced response collapse* has been ascribed to shallow template following, but its mechanism remains unclear. We show that lexical bans are only one of many triggers: across seven models, everyday constraints on tone, formatting, and delivery cut comprehensiveness by up to 45%, even though the model obeys them. Controlled ablations identify the trigger as *form primers*, phrases in the prompt that refer to the form of the response rather than its content. Studying each post-training stage of the open `OLMo` family, we trace the collapse to reward-based post-training rather than template following: it emerges at preference optimization and is largest after reinforcement learning from verifiable rewards, two stages whose rewards do not measure content. A *single direction* in the model's activations controls the collapse: adding it to unconstrained prompts induces collapse, and subtracting it from constrained prompts restores most of the lost content with the constraint still obeyed, mitigating the collapse at inference time. Because pointwise judges miss about half of this loss, we release a 160-question benchmark that scores every answer against the same model's unconstrained answer.

1 INTRODUCTION

Instruction-tuned language models are now the first point of contact for everyday questions, through chat assistants, search results, and customer support. In these settings, the questions often arrive with instructions about the form of the reply: keep the tone professional, do not use em dashes, etc. We call such an instruction a *constraint*, meaning a request about the *form* of an answer added to a task's prompt. Following such constraints is a headline capability of instruction-tuned models (Ouyang et al., 2022) and the target of widely used benchmarks (Zhou et al., 2023b). Instruction-following work primarily evaluates whether constraints are satisfied, and in this work, we further the study on how obeying such instructions often comes at a substantial cost to the quality of the response.

Constraints typically concern the form of the answer rather than its content, so a reasonable expectation is that they largely preserve response substance. Yet, Baghaei Potraghloo et al. (2026) showed that one added constraint causes up to 48% less response substance. Figure 1 shows an example detailed/formatted model response that becomes dramatically less detailed when given a simple constraint. This effect is called *constraint-induced response collapse* and was established mainly through token-level bans. The effect was attributed to *template dependence*: a banned token disables a learned response format which degrades the response. In this paper, we find that the collapse occurs even when an irrelevant, *no-op* constraint is present, e.g., "do not talk about kittens." The *no-op* constraints are not violated and should be ignored, yet result in up to a 30% drop in comprehensiveness on `Qwen3-4B-Instruct` (see Figure 2). Template dependence does not explain such collapse, since no templates can be learned in correlation to no-op constraints.

Two things make this problem relevant to deployed production models. First: the collapse is silent and easy to miss since the constrained response looks reasonable on its own. Both the training rewards and standard benchmarks score answers (Zhou et al., 2023b) see this response in isolation, so they have no signal that anything is wrong, but the collapse is clear once the constrained answer is

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.

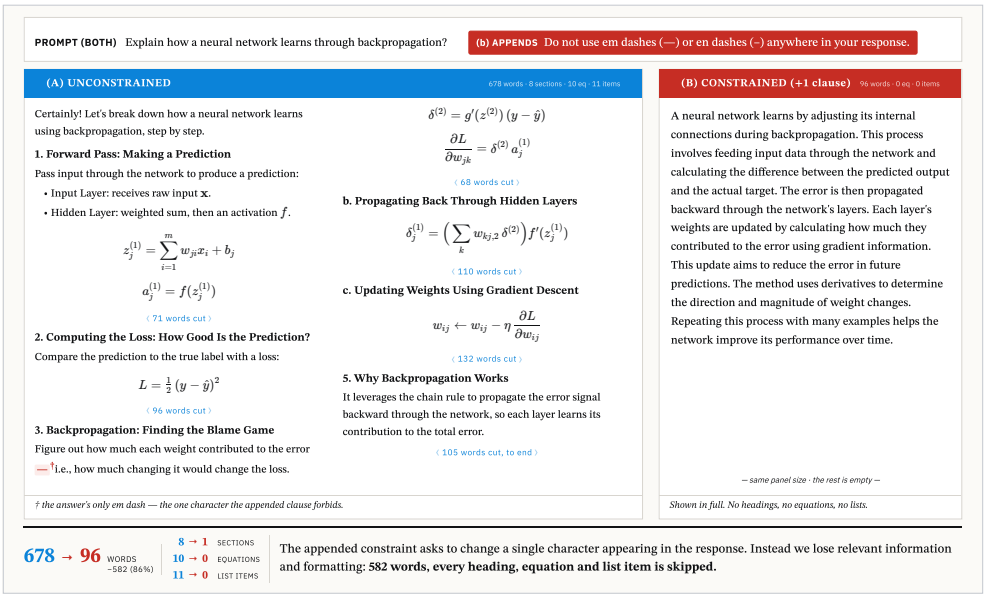

Figure 1. Constraint-induced response collapse (same model, same prompt). **Left:** without a constraint, the answer is formatted and detailed. **Right:** adding one em-dash constraint cuts it to a single flat paragraph. The constrained answer keeps 1 of 8 sections, none of the 10 equations or 11 list items, and 96 of 678 words.

compared with the unconstrained. Second: prior work mostly studies this in the context of lexical character bans, but there are many additional, realistic constraints that trigger this collapse, silently.

Contributions. In this paper, we identify realistic constraints that trigger the collapse and perform control experiments to characterize and isolate them. This includes no-op constraints that should have been ignored, but still trigger the collapse. Across seven instruction-tuned models, comprehensiveness drops by as much as 45% (Table 1). We also verify that the content is withheld rather than lost, since a single follow-up asking for a complete answer restores it (Section 2.1).

We show that the judges used in training and in instruction-following benchmarks report about half the drop that a side-by-side comparison finds, because an omission cannot be seen in a single answer. We therefore release a benchmark that scores every answer with and without a constraint: 160 questions, each paired with 11 everyday constraints on tone, formatting, language, and delivery, scored by a pairwise judge and by a factual-coverage count that extra words cannot inflate (Section 2.1).

We provide empirical evidence that a single direction in the residual stream, the difference in mean activations between constrained and unconstrained prompts, controls the collapse and can be used to mitigate it. Adding that direction induces the collapse on unconstrained prompts, and subtracting it reverses the collapse on constrained prompts, on two model families, `Qwen3-4B-Instruct` and `Llama-3.1-8B-Instruct`, without cost to constraint compliance (Section 3).

Finally we trace the emergence of collapse to the reward-based stages of post-training. It is not tied to one training method: it shows up whenever training rewards something other than the substance of the answer, such as its length, its formatting, or whether it obeys constraints. It first appears at the preference-optimization stage, where the reward favors long, structured answers and makes that the model's only learned way to be thorough, so a constraint that steers it toward a shorter, plainer, or less structured reply loses the content along with the style; and it grows at the reinforcement-learning stage, whose instruction-following reward is a pass/fail check on each constraint that even a short, unhelpful answer can satisfy. In OLMo training, the drop is near zero after SFT and reaches 33% after RLVR (Table 5, Section 4).

1.1 Related Work

Constraint following and its costs. Instruction-following benchmarks score whether a stated constraint is satisfied (Zhou et al., 2023b; Jiang et al., 2024; Qin et al., 2024; Pyatkin et al., 2025) but

typically do not measure the output quality degradation. Recently, Baghaei Potraghloo et al. (2026) measured this cost for lexical bans, but we further show that no-op constraints still cost 15 to 38% of the content (Table 2). Format restrictions and irrelevant clauses are known to lower accuracy on tasks with a right answer (Tam et al., 2024; Shi et al., 2023; Mirzadeh et al., 2025), and hard constrained decoding can exclude the intermediate tokens a derivation needs (Banerjee et al., 2025); our constraints restrict neither the token set nor the grammar, and we measure what an open-ended answer leaves out, which an accuracy check cannot see.

Judge protocols. We adopt the side-by-side protocol of Zheng et al. (2023); because such judges reward length and style (Dubois et al., 2024; Feuer et al., 2025), we also report a length-discounted variant (Table 3). We build on atomic-fact evaluation (Min et al., 2023). Prior protocols score a response against an external source or a competing model; we score it against the answer the same model gives without the constraint, so it degrades only by dropping content the model supplied.

Steering. A single direction in the residual stream can carry a behavior, as shown for sentiment (Turner et al., 2023) and refusal (Arditi et al., 2024). Closest to our work, Stolfo et al. (2025) compute instruction vectors as we compute our collapse direction, from the difference of mean activations with and without an instruction, and steer with them to improve adherence, adding them for most constraints and subtracting word-inclusion vectors to enforce exclusion. With a pointwise judge, they also find that constraints lower response quality, and they treat this as a trade-off between adherence and comprehensiveness, since satisfying a constraint "may lead to less comprehensive responses". We show that pointwise judges like theirs capture only about half of the loss that a pairwise judge detects (Section 2.1). More importantly, we demonstrate that the trade-off is not inherent: no-op constraints that require nothing still cost content, a follow-up request recovers it with the constraint in force, and subtracting our direction while the constraint stays in the prompt restores most of it, with compliance within about two points of the unsteered answers. Finally, we trace the collapse's root cause, which they do not examine, to reward-based post-training (Section 4).

Post-training rewards. Optimizing a proxy reward too hard moves a model away from the behavior the proxy stands for (Gao et al., 2023). In the pipeline we study (Team Olmo, 2025), neither preference optimization (Rafailov et al., 2023) nor the instruction-following verifier of reinforcement learning from verifiable rewards (Lambert et al., 2025) measures content. Our effect is neither a preference for length (Stiennon et al., 2020; Park et al., 2024) nor likelihood displacement (Razin et al., 2025): it survives at matched length (Section 2.1), standard and length-normalized DPO (Meng et al., 2024) lose the same content at matched quality (Table 6), and it depends on the constraint being present rather than a property of the trained model (Table 2). Appendix K extends this discussion.

2 Diagnosing the Collapse

Constraint delivery. For a base question q and a single constraint c, the constrained prompt is c concatenated to q (in the user turn, unless noted).The unconstrained prompt q serves as the baseline. Appendix F lists every constraint we use verbatim, including those used only for ablations.

Evaluation. Our primary metric is *comprehensiveness*: substantive content conveyed about q. We measure it with an LLM-as-a-judge evaluator in a *pairwise* protocol (Zheng et al., 2023), in which both answers are shown side by side and each is scored from 1 to 10; the pairwise protocol is required because a pointwise judge does not have a baseline for missed content (Section 2.1). We report the relative change Δ of the constrained answer against its unconstrained baseline, and the *unconstrained win-rate*, which is the fraction of questions in which the unconstrained answer is judged more comprehensive. We also report *factual coverage*, which extracts the atomic claims of the unconstrained answer and counts how many of them the constrained answer still conveys. Coverage cannot be raised by writing more words, so it decouples lost content from lost length and verbosity. Every score in the paper (comprehensiveness, factual coverage, and instruction-following compliance) is obtained via `Claude Opus 4.8` (Anthropic, 2026), unless otherwise specified. Appendix A gives both judge prompts, the scoring rules, and the formulas for Δ and the win-rate.

The benchmark. We release a benchmark that pairs 160 questions with the fixed eleven-constraint panel shown in Figure 2 (the figure also shows professional tone) and the pairwise comprehensiveness

Model	Δ comp. (%)	Unconstr. win (%)	Word Δ (%)
Qwen-2.5-7B	−44.7	98.8	−59.9
Qwen3-30B-A3B	−37.9	95.0	−56.9
Llama-3.3-70B	−34.7	91.6	−43.7
Olmo-3-7B	−33.4	93.0	−45.6
Qwen3-4B	−29.0	93.9	−41.7
Llama-3.1-8B	−21.6	83.0	−17.5
Mistral-7B	−17.4	78.5	−12.3

Table 1. Collapse across seven instruction-tuned models, ordered by the size of the loss. Δ comp. is the change in comprehensiveness against the same model's unconstrained answer; Unconstr. win is the share of pairs in which the judge prefers the unconstrained answer; Word Δ is the change in length. Every model loses comprehensiveness (17% to 45%), and the unconstrained answer is preferred over the constrained answer in a majority of pairs (78% to 99%).

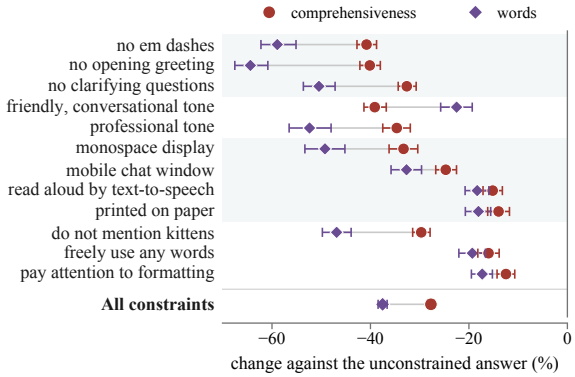

Figure 2. Per-constraint collapse on the 160-question benchmark (`Qwen3-4B-Instruct`, 1,920 pairs: the eleven benchmark constraints plus professional tone; *All constraints* averages the twelve rows). Red: change in comprehensiveness; violet: change in word count. Bands separate restriction, register, realization, and no-op constraints, bars indicate 95% confidence interval. Every constraint costs content, from −12.5% (pay attention to formatting) to −40.8% (no em dashes).

protocol above. The questions come largely from AlpacaEval (Li et al., 2023) and MT-Bench (Zheng et al., 2023), with 40 written specifically for this work, and are chosen so that answers have room to be thorough and their quality can be measured (Figure 5 gives the composition). The eleven constraints cover the four kinds of form primers defined in Section 2.1 (restriction, register, realization, no-op), and results are reported with the metrics above: Δ, the unconstrained win-rate, and factual coverage (Appendix B). Because every score is measured against the answer the same model gives without the constraint, the benchmark registers what an answer leaves out, which pointwise instruction-following benchmarks miss.

Models and code. We report results on seven instruction-tuned models across the `OLMo, Qwen, Mistral and LLaMA` families. Every experiment runs on open-weight models and public training code, so each result can be checked independently. The causal steering experiment reads internal activations, which requires open weights, and the training analysis reads the reward function and the optimizer settings, which requires public code. The `OLMo` family releases its supervised fine-tuning code (`OLMo-core`), its reinforcement-learning code (`open-instruct`), and a checkpoint after each post-training stage, so it is the family we use to trace the origin of the collapse in Section 4.

2.1 Measuring the Collapse and Isolating the Trigger

Table 1 shows that all seven models lose a large share of response content once a constraint is added, from 17% on the mildest to 45% on the worst. The constraints that trigger it are not adversarial but everyday instructions about formatting, tone, and output channel, the kind that arrive with ordinary requests from a system prompt or from a user. Not a single constraint in the panel leaves the answer intact (Figure 2). What the model drops is *content* rather than just verbosity, and the drop persists even when length between the two responses matches. In the matched-length case, an answer to "describe the process of DNA replication in cells" given one appended clause asking for a warm, conversational tone comes back at 612 words against the unconstrained 614, fluent and obedient, yet conveys only 5 of the 20 facts the unconstrained answer states. What it drops is relevant to answering the question: how the two strands are copied, and how errors are found and corrected.

The *template dependence* account of prior work (Section 1; Baghaei Potraghloo et al., 2026) is right about one case: a ban on em dashes removes a token that the model's usual format relies on, and it produces the largest collapse in Figure 2. However, it misses the rest, because many prompts that trigger the collapse block no token at all. In fact, what those prompts have in common is the thing they refer to, what we call a *form primer*, and the rest of this section explores this more precisely.

Table 2. Form priming hypothesis ablation on the final checkpoint of two models, 40 questions per row; only the appended sentence changes (abridged; verbatim in Appendix F). R1–R3 mark the property each alternative explanation needs: the sentence is binding, i.e. a typical answer would violate it (R1), is phrased as an order (R2), or hints at a shorter answer (R3). A row that lacks the property and still collapses rules that alternative out; R3 holds in no row. The shaded *Form?* column marks form primers (Section 2.1): every one collapses the answer, while the two controls below the rule, which are not about the form, do not, ruling out R4 (that any added sentence would do). Δ is the change in comprehensiveness; the unconstrained win rate is in parentheses.

Appended sentence (abridged)	R1 Binding?	R2 Order?	R3 Shorter?	Form?	OLMo-final Δ (win%)	Qwen3-4B Δ (win%)
Don't use em or en dashes	✓	✓	✗	✓	−51 (100)	−42 (100)
Don't discuss kittens or cats	✗	✓	✗	✓	−23 (95)	−30 (100)
Freely use any words	✗	✓	✗	✓	−38 (100)	−15 (80)
Will be read aloud by TTS	✗	✗	✗	✓	−25 (98)	−20 (95)
Will be shown in monospace	✗	✗	✗	✓	−20 (90)	−33 (98)
Mind formatting and punctuation	✗	✓	✗	✓	−22 (93)	−15 (83)
Italicize each heading	✓	✓	✗	✓	−28 (90)	−30 (98)
Be thorough, include all details	✗	✓	✗	✗	+18 (8)	+37 (3)
Address it in the best possible way	✗	✓	✗	✗	−5 (53)	−3 (48)

What counts as a form primer. A sentence *primes the form of a response*, and is a *form primer*, when the thing it talks about is one of the properties of the response as a piece of text rather than of its content: such as the characters it uses, its layout and formatting, its register and tone, and the medium in which it will be delivered. "Your response will be displayed in a monospace font" talks about how the answer will be shown, so it is a form primer; "be thorough and include all relevant details" talks about what the answer should contain, so it is not. We use the name *form priming* for the effect that such a sentence has, because Section 3 shows that it moves the model's internal state while the model is still reading the prompt. The pointers we study fall into four groups, and the benchmark panel (Figure 2) uses the same four: *restrictions*, which forbid or require something about the form; *register*, which asks for a particular tone or reading level; *realization*, which states how the answer will be displayed or delivered; and *no-op constraints*, which point at the form without restricting anything that is present in an answer.

Ruling out four alternate hypotheses. A sentence that points at the form usually does three other things at once: it forbids something the answer would otherwise have done, it is phrased as an order, and it hints that a shorter reply is safer. Any of these could be the real cause, which gives four alternatives: the collapse requires the sentence to forbid something the answer needs (R1), to be an order (R2), or to hint at a shorter answer (R3); or any added sentence would cause it, no matter what it says (R4). Table 2 removes them one at a time, leaving form priming as the likely explanation.

None of the four hypotheses survive (Table 2). R1: a ban on something no answer would contain ("Do not discuss kittens, cats or any feline animals") and a permission that forbids nothing ("Freely use any words in your response") still cost 15 to 38% of the content. R2: plain statements of fact that give no order, that the answer will be read aloud or shown in a monospace font, cost 20 to 33%. R3: no sentence in the table hints at a shorter answer; "Pay attention to your formatting and punctuation as you write" asks for no change at all and still costs 15 to 22%, and "Italicize each section's heading", which asks for more text rather than less, still costs 28 to 30%. R4: a sentence about content rather than form ("Be thorough and include all relevant details") raises comprehensiveness by 18 to 37%; because that sentence does ask for more detail, a second control that does not ("Address the question in the best possible way") leaves it nearly unchanged (−5 and −3%), so the cost comes from the reference to the form and not from adding a sentence. The effect also holds across constraints and models: ten of the eleven deployment constraints collapse on both families, which rank them almost identically (Spearman 0.77).

Counting facts instead of scoring answers. The comprehensiveness score in Figure 2 is a judge's opinion on a 1–10 scale. Measuring factual coverage adds a length agnostic measure to remove the confounding nature of comprehensiveness and length. We report the factual coverage count of Section 2 on 1,760 pairs from the eleven benchmark constraints: a judge lists the atomic claims of

the unconstrained answer, about 19 per answer, and marks which of them the constrained answer still conveys, in any wording. Averaged over the eleven constraints, a constrained answer conveys 36.1% of those facts, against 58.0% for a second unconstrained answer to the same question, which is the level an answer reaches when nothing is withheld. Every constraint, the no-op constraints included, falls below that reference, and the attention-to-formatting sentence keeps the most (46.6%), as in Figure 2. The missing facts are not only minor ones: the judge also grades each claim from 0 (trivial) to 3 (essential to a complete answer), and essential claims are dropped 42% of the time. Appendix B gives the procedure, the reference, and the numbers for every constraint.

Pointwise judges see half of the full story. The judge used during training and the judge used for evaluation are both pointwise: each sees a single answer and asks only whether it addresses the question, so a competent but terse answer passes and the omission goes undetected. Table 3 re-scores the same answers on two models under three protocols: the pointwise judge, our pairwise judge, and a pairwise judge that is also told to ignore length, formatting, and style and to score only the distinct substantive points (Appendix A). Pointwise scoring reports about half the loss that the pairwise protocol reports on both models: -13.4% against -29.0% on `Qwen3-4B-Instruct`, and -14.3% against -21.6% on `Llama-3.1-8B-Instruct`. With length explicitly discounted the loss is still -26.6% and -19.1%, so most of it is not an artifact of the shorter answer. Standard instruction-following and quality evaluations do not measure this loss, and on short-answer , they reward it, which is why an effect of this size has gone unnoticed.

The content is withheld instead of being lost. The model has not lost the ability to give a full answer: it still holds the capacity to output what it left out, and one short request gets it back. On `Qwen3-4B-Instruct`, over the eleven deployment constraints and 40 questions, we take the constrained answers and, with the constraint still in place, send a second message that says only "Please give the complete answer." Under the constraint, comprehensiveness is 31.4% below the unconstrained answer, which wins 96% of the comparisons; after the request, the new answer is 3.6% above it and the unconstrained answer wins only 35%, so the new answer is as complete as the one given with no constraint at all. The new answer is also about 48% longer, as expected if the missing material is now written out (Appendix E, Table 11). The recovery is not the effect of taking a second turn: a second message that only says "Please continue." does not recover the content, so what matters is that the model is asked for a complete answer. The recovery is also not equally strong for every constraint. It is largest for the constraints that cost the least to begin with. The level reached after the request tracks the size of the original drop closely (r=0.89): eight of the eleven constraints come back to within 7% of the unconstrained answer, while the three that cost the most, a ban on em dashes, a friendly tone, and a ban on conversational openers, stay short of it.

3 Mitigating the Collapse through Linear Activation Steering

Section 2.1 showed that the model still has the knowledge it omits from its response, and as we discuss here, a single linear activation intervention (a "recovery vector") can either induce or recover collapse. Constraints on tone, formatting, or language, and even a permission that forbids nothing, all produce the same collapse, consistent with each one moving the prompt's activation along this single axis. Moving the model along the direction at inference time turns comprehensiveness down or brings it back with no retraining, which gives a practical way to study the effect and to reduce it.

Calculating the steering direction. Constrained and unconstrained prompts fall into distinct regions of activation space. We run the experiment on two instruction-tuned models from different families, `Qwen3-4B-Instruct` and `Llama-3.1-8B-Instruct`. For each prompt we read the residual-stream activation at a single layer (a mid-depth layer for `Qwen3-4B`, layer 8 of 32 for `Llama-3.1-8B`) over the last prompt token, and define the *collapse direction* v as the mean activation over constrained prompts minus the mean over their unconstrained counterparts. This is the difference-of-means construction that isolates the refusal direction (Arditi et al., 2024); it uses the prompts alone and involves no optimization or fitting.

The intervention. During generation we add γv to the residual stream at that layer and sweep the strength γ (Table 4, Figure 3). *Inducing*: adding $+\gamma v$ to an unconstrained prompt, whose text asks for nothing and names no constraint, drops the answer to the collapsed level: a short reply with little detail,

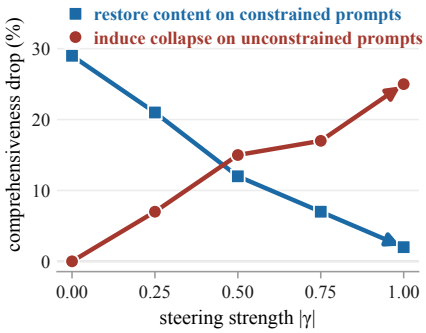

Figure 3. One direction determines comprehensiveness (`Qwen3-4B-Instruct`). Steering along the collapse direction v swaps the two conditions as $|\gamma|$ grows: $+\gamma v$ pushes unconstrained answers into collapse, and $-\gamma v$ uncollapses. A norm-matched random direction does neither (Table 4).

Judge protocol	Base	Constr.	Δ (%)	Win (%)
`Qwen3-4B-Instruct`				
Pointwise (training)	8.41	7.28	−13.4	63.1
Pairwise (ours)	8.68	6.16	−29.0	93.9
Pairwise, length disc.	8.53	6.26	−26.6	88.5
`Llama-3.1-8B-Instruct`				
Pointwise (training)	6.53	5.59	−14.3	63.6
Pairwise (ours)	7.74	6.07	−21.6	83.0
Pairwise, length disc.	7.58	6.13	−19.1	76.4

Table 3. The same answers under three judge protocols: *pointwise*, as used in training; *pairwise*, ours (Table 1); and pairwise with an instruction to disregard length. Base and Constr. are mean comprehensiveness (1–10) of the unconstrained and constrained answers; Win is the share of pairs won by the unconstrained answer.

Table 4. Steering with the collapse direction v on two model families. Indented rows steer the prompt type above them (γ=1 for `Llama-3.1-8B`); r is a norm-matched random direction. Comp. is mean comprehensiveness (1–10) and Words refers to response length; Δ (%) is relative to the unconstrained answer; Complies (%) is the share of responses judged to obey the constraint; PPL is perplexity under the unsteered model.

	`Qwen3-4B-Instruct`					`Llama-3.1-8B-Instruct`				
	Comp.	Words	Δ	Complies	PPL	Comp.	Words	Δ	Complies	PPL
Unconstrained	8.68	568	ref.	—	1.40	7.74	471	ref.	—	1.20
$+\gamma v$ (induce)	6.50	384	−25	—	1.47	6.53	399	−14	—	1.25
$+\gamma r$ (random)	8.62	549	0.7	—	1.54	6.95	477	−3	—	1.20
Constrained	6.17	331	−29	90.5	1.40	6.07	389	−22	97.0	1.25
$-\gamma v$ (rescue)	8.48	591	−2	88.2	1.50	6.83	502	−8	97.3	1.25

as if a constraint were present. *Rescuing:* subtracting γv from a constrained prompt brings the answer back. On `Qwen3-4B` it returns to a statistical tie with the unconstrained baseline (−2%, Table 4) and recovers the content that the constraint had removed, while the constraint is still in the prompt and still obeyed; the effect grows monotonically with γ (Figure 3), and a norm-matched random direction barely moves comprehensiveness. The construction carries over to `Llama-3.1-8B` at γ=1 (Table 4, bottom): adding v costs 14 points of comprehensiveness against 3 for the norm-matched random direction, and subtracting it recovers 14 of the 22 points that the constraint removes. The direction causes the collapse in both directions rather than merely tracking it, and the random control rules out an explanation based on the size of the perturbation alone.

The rescue does not cost compliance or fluency. A rescue would prove nothing if it recovered content by dropping the constraint or by degrading the text, so we check both (details in Appendix C). A single-response judge rates whether each response obeys its constraint: compliance under the rescue is about the same as in the unsteered constrained condition, 88.2% against 90.5% on `Qwen3-4B` and 97.3% against 97.0% on `Llama-3.1-8B`, over 440 responses each (Table 4). Mean perplexity under the unsteered model is also about the same, 1.50 against 1.40 on `Qwen3-4B` and 1.25 in both conditions on `Llama-3.1-8B`. The model still follows the instruction and stops withholding content, and the recovered content reads as ordinary text.

4 The Upstream Root Cause: Reward-Based Post-Training

We study the impact of different training stages on the `OLMo` family, since they release a checkpoint after each stage and publishes the training code. We find that the collapse is absent after supervised fine-tuning (SFT) and appears during the two stages that follow, preference optimization (DPO) and

Table 5. Collapse on OLMo post-training stage checkpoints, along with the reward optimized. Δ is against each stage's unconstrained answer, over 11 constraints and 40 questions (440 pairs per stage), one judging run per stage.

Stage	Δ comp. (%)	Unconstr. win (%)	Reward being optimized	Measures content?
SFT	≈ 0	47.0	max-likelihood on demonstrations	n/a
DPO	-26.5	81.6	implicit: longer, more formatted wins	✗ length/format
RLVR (final)	-33.4	93.0	binary constraint-satisfaction (β=0)	✗ no

Table 6. Collapse tracks optimization strength, not loss form: lower KL β gives richer answers and more collapse; at equal quality, standard and length-normalized DPO collapse alike.

Objective	KL β	Unconstrained comp.	Δ comp. (%)
Standard DPO	1.0	5.2	$+3$
Standard DPO	0.1	6.9	-4
Standard DPO	0.03	8.1	-21
Length-normalized DPO (OLMo)	5	8.2	-22
RLVR, released checkpoint	0	8.6	-33

reinforcement learning from verifiable rewards (RLVR). Table 5 reports collapse stage by stage and mentions the reward each stage optimizes read from the public code. Neither the DPO nor the RLVR reward measures content. Three findings follow: the collapse is a general consequence of that blind spot, its size depends on how far the policy is optimized rather than on the particular loss, and the two stages drop content for different reasons.

Optimization strength sets the size of the collapse. SFT imitates thorough demonstrations and does not collapse; the collapse appears at DPO ($\Delta \approx -27\%$) and is largest after RLVR ($\Delta \approx -33\%$). To separate the DPO *objective* from the optimization itself, we retrain from the released SFT checkpoint with standard (sum-based) DPO and vary the KL coefficient β, which controls how far the policy may move from SFT (Table 6). As β falls, the unconstrained answer becomes richer and the collapse grows with it. At β=1.0 the policy barely leaves SFT and does not collapse. At β=0.03 it reaches the quality of the length-normalized `dpo_norm` that OLMo uses and collapses just as hard (-21% against -22%), so length normalization does not produce the collapse. RLVR runs at β=0, the far end of this curve, consistent with its collapse being the largest of the three stages. The collapse follows from optimizing the policy away from SFT against a content-blind reward.

DPO: thoroughness bound to a single elaborated answer. DPO has no explicit reward; it amplifies whatever separates preferred from dispreferred responses in its data. In OLMo's 6,750 judge-labeled `Dolci` preference pairs, the preferred answer is the longer one in 58% of cases, and among pairs matched for length it is the more heavily formatted one (44% against 27%, with more section headers, bullets, and bold). DPO therefore leaves the model with one default way to answer well: at length, laid out in sections, with the points enumerated and the key terms marked. In the trained model these traits are no longer separable: across our constraints, content loss, length loss, and structure loss rise and fall together (pairwise $r \approx 0.7$–0.8), so none of them can be singled out as the cause. Unprompted, the model has no second way of being thorough (asked for a complete answer, `Qwen3-4B-Instruct` recovers most of the content; Section 2.1), and that is what makes the collapse possible. Pushed off its default answer, it does not re-express the same material in the permitted form; it returns a shorter and flatter answer that makes fewer points.

RLVR: a binary compliance reward. Following DPO, the cost of a constraint still depends on the constraint itself, while the RL stage adds a cost that does not have this dependency. This is explained by the reward: OLMo's `IFEvalVerifier` returns the fraction of the prompt's constraints that pass a binary programmatic check, with no term for length or content, so a one-sentence answer that satisfies every check scores a perfect 1.0, and by being shorter, has fewer places to fail a check (see Appendix G). Under any constraint, then, the reward prefers the shortest compliant answer, and that pressure should not depend on which constraint is present. On the released DPO and RLVR checkpoints (Appendix H), the average drop grows from 22.9% to 31.7% on one paired judging run, and RLVR subtracts a near-constant 8.8 percentage points of comprehensiveness Δ from whatever DPO left, with a standard deviation of 3.2 across the eleven constraints, and regressing the RLVR values on the DPO values gives a slope of 0.96 (95% CI 0.80 to 1.11): the stage applies an offset rather than an amplification. One case separates the two readings on its own: "your response will be printed on paper" gains content after DPO ($+2.9\%$) and loses it after RLVR (-6.0%), and no amplification of an existing effect can reverse a sign. Beyond DPO's dependence on the elaborate answer style, RLVR adds a general, constraint-blind pressure toward the minimal compliant answer.

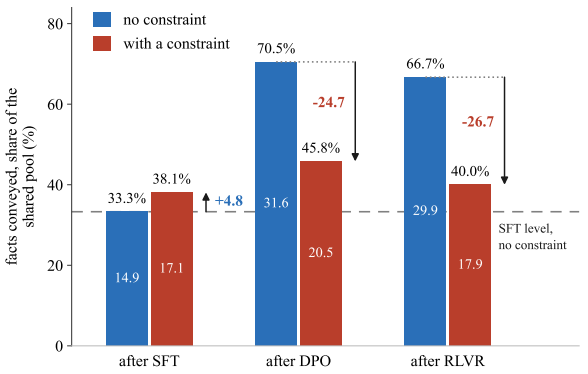

Figure 4. Facts conveyed at each stage, with and without a constraint, as a share of one pool per question built from all three stages' unconstrained answers. Inside bars: mean facts per answer over 40 questions; constrained bars also average the eleven constraints.

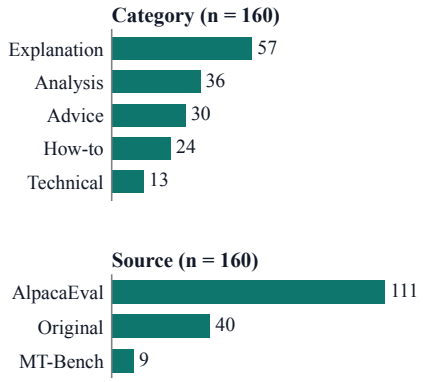

Figure 5. The 160-question benchmark by category and source. Questions are drawn from AlpacaEval (Li et al., 2023) and MT-Bench (Zheng et al., 2023), with 40 written for this work.

Why a grounded reward spares math. A reward that checks the answer itself, as for math or code, should not produce the collapse because its reward is *grounded*: the content the user wants is what the model must produce to pass, so optimization protects it. We test this for math on AIME 2026 with `Qwen3-4B-Thinking-2507`, a reasoning model trained with a verifiable reward (Appendix J). The form primer still reaches the model: a plain-language constraint shortens the reasoning trace by 13.7% and a friendly tone by 9.7%, while the no-op "freely use any words in your response" (Table 2) leaves it unchanged. The graded answer does not measurably move: no constraint shifts accuracy by more than 4 points, and every interval includes zero. What matters is whether the reward measures the content the user wants: a verifiable reward does, an instruction-following reward does not, and a form priming constraint is where the latter stops tracking the content.

Factual coverage. The pattern also holds when we count facts instead of scoring answers, which puts it on an absolute scale. For each question we pool the facts stated by the unconstrained answers of all three `OLMo` stages, about 45 per question, and score every response against that one pool (Figure 4). Post-training roughly doubles how much of the pool the model conveys, from 33.3% after SFT to 70.5% after DPO, and a single constraint gives most of that back: after RLVR the constrained answer conveys 40.0%, only a few points above the supervised model with no constraint at all. In counts, post-training adds about 15 facts per answer, and a constraint keeps only 3 of them; Appendix A breaks this down into the facts a constraint keeps, loses, and adds.

5 Conclusion

Constraint-induced response collapse is a fragility in instruction-tuned models: one constraint can make a model comply while withholding much of its content. We isolate the triggering subset of constraints to be *form primers*: constraints that dictate the form of the response rather than the content. We show that this collapse occurs even with no-op constraints that a model's response would never violate in the first place. We then identified that a single internal direction causally controls the withheld content, and can be used to both induce collapse in unconstrained prompts, and rescue constrained prompts from collapse. Finally, we trace the origin of the collapse by studying the different stages of the OLMo family, and find that it originates in reward-based post-training. The rewards in these stages do not measure content, through a preference for the formatted style at the preference-optimization stage and a binary compliance reward at the reinforcement-learning stage. Both mechanisms share one fix, making retained content the quantity that the reward optimizes: a reward that scores how much of a comprehensive reference answer's substance the response conveys, blind to length, and treats constraint satisfaction as a floor rather than a term traded against content. This reward passes two checks that need no training (Appendix I); the decisive test, a short reinforcement-learning run with it, is left to future work. All of our results use open models and public code, and we additionally release a benchmark that measures the effect.

AI USE STATEMENT

`Claude Opus 4.8` and `Sonnet 5` were used as judges to produce all scores reported in the manuscript (see Appendix A) and to help draft and polish the text. The authors designed all experiments and take full responsibility for the entirety of the content.

ETHICS STATEMENT

The authors have read and adhere to the ICLR Code of Ethics. This work involves no human subjects and no personal data: the questions come from the public AlpacaEval and MT-Bench sets and from 40 questions written by the authors, and the constraints are everyday instructions on form (Appendix F). The work measures and reduces a loss of helpfulness in instruction-tuned models.

REPRODUCIBILITY STATEMENT

Every experiment uses open-weight models, and the post-training analysis uses the released `OLMo` checkpoints and public training code (Section 2). Appendix F gives every constraint verbatim, Appendix A the judge prompts and scoring rules, Figure 5 the composition of the released benchmark, and Appendix C the steering checks.

REFERENCES

Anthropic. Claude Opus 4.8 System Card. Anthropic, 2026. `https://www.anthropic.com/claude-opus-4-8-system-card`.

Arditi, A., Obeso, O., Syed, A., Paleka, D., Panickssery, N., Gurnee, W., and Nanda, N. Refusal in Language Models Is Mediated by a Single Direction. In *Advances in Neural Information Processing Systems (NeurIPS) 37*, 2024. arXiv:2406.11717.

Baghaei Potraghloo, E., Azizi, S., Kundu, S., and Pedram, M. One Token Away from Collapse: The Fragility of Instruction-Tuned Helpfulness. arXiv:2604.13006, 2026.

Banerjee, D., Suresh, T., Ugare, S., Misailovic, S., and Singh, G. CRANE: Reasoning with Constrained LLM Generation. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, pp. 2836–2857, 2025.

Dubois, Y., Galambosi, B., Liang, P., and Hashimoto, T. B. Length-Controlled AlpacaEval: A Simple Way to Debias Automatic Evaluators. In *Conference on Language Modeling (COLM)*, 2024. arXiv:2404.04475.

Feuer, B., Goldblum, M., Datta, T., Nambiar, S., Besaleli, R., Dooley, S., Cembalest, M., and Dickerson, J. P. Style Outweighs Substance: Failure Modes of LLM Judges in Alignment Benchmarking. In *International Conference on Learning Representations (ICLR)*, pp. 11552–11582, 2025. arXiv:2409.15268.

Gao, L., Schulman, J., and Hilton, J. Scaling Laws for Reward Model Overoptimization. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pp. 10835–10866, 2023. arXiv:2210.10760.

Hu, Z., Song, L., Zhang, J., Xiao, Z., Wang, T., Chen, Z., Yuan, N. J., Lian, J., Ding, K., and Xiong, H. Explaining Length Bias in LLM-Based Preference Evaluations. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 6763–6794, 2025. arXiv:2407.01085.

Jiang, Y., Wang, Y., Zeng, X., Zhong, W., Li, L., Mi, F., Shang, L., Jiang, X., Liu, Q., and Wang, W. FollowBench: A Multi-level Fine-grained Constraints Following Benchmark for Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4667–4688, 2024. arXiv:2310.20410.

Kirk, R., Mediratta, I., Nalmpantis, C., Luketina, J., Hambro, E., Grefenstette, E., and Raileanu, R. Understanding the Effects of RLHF on LLM Generalisation and Diversity. In *International Conference on Learning Representations (ICLR)*, pp. 20620–20653, 2024. arXiv:2310.06452.

Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., Brahman, F., Miranda, L. J. V., Liu, A., Dziri, N., Lyu, S., Gu, Y., Malik, S., Graf, V., Hwang, J. D., Yang, J., Le Bras, R., Tafjord, O., Wilhelm, C., Soldaini, L., Smith, N. A., Wang, Y., Dasigi, P., and Hajishirzi, H. Tülu 3: Pushing Frontiers in Open Language Model Post-Training. In *Conference on Language Modeling (COLM)*, 2025. arXiv:2411.15124.

Li, X., Zhang, T., Dubois, Y., Taori, R., Gulrajani, I., Guestrin, C., Liang, P., and Hashimoto, T. B. AlpacaEval: An Automatic Evaluator of Instruction-following Models. 2023. https://github.com/tatsu-lab/alpaca_eval.

Lin, B. Y., Ravichander, A., Lu, X., Dziri, N., Sclar, M., Chandu, K., Bhagavatula, C., and Choi, Y. The Unlocking Spell on Base LLMs: Rethinking Alignment via In-Context Learning. In *International Conference on Learning Representations (ICLR)*, pp. 24907–24933, 2024. arXiv:2312.01552.

Lu, J., Li, J., An, S., Zhao, M., He, Y., Yin, D., and Sun, X. Eliminating Biased Length Reliance of Direct Preference Optimization via Down-Sampled KL Divergence. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1047–1067, 2024. arXiv:2406.10957.

Meng, Y., Xia, M., and Chen, D. SimPO: Simple Preference Optimization with a Reference-Free Reward. In *Advances in Neural Information Processing Systems (NeurIPS) 37*, 2024. arXiv:2405.14734.

Min, S., Krishna, K., Lyu, X., Lewis, M., Yih, W., Koh, P. W., Iyyer, M., Zettlemoyer, L., and Hajishirzi, H. FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. arXiv:2305.14251.

Mirzadeh, I., Alizadeh-Vahid, K., Shahrokhi, H., Tuzel, O., Bengio, S., and Farajtabar, M. GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models. In *International Conference on Learning Representations (ICLR)*, pp. 94743–94765, 2025. arXiv:2410.05229.

Team Olmo. Olmo 3. arXiv:2512.13961, 2025.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS) 35*, 2022. arXiv:2203.02155.

Pan, A., Bhatia, K., and Steinhardt, J. The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2201.03544.

Park, R., Rafailov, R., Ermon, S., and Finn, C. Disentangling Length from Quality in Direct Preference Optimization. In *Findings of the Association for Computational Linguistics (ACL Findings)*, pp. 4998–5017, 2024. arXiv:2403.19159.

Pyatkin, V., Malik, S., Graf, V., Ivison, H., Huang, S., Dasigi, P., Lambert, N., and Hajishirzi, H. Generalizing Verifiable Instruction Following. In *Advances in Neural Information Processing Systems (NeurIPS) 38, Datasets and Benchmarks Track*, 2025. arXiv:2507.02833.

Qin, Y., Song, K., Hu, Y., Yao, W., Cho, S., Wang, X., Wu, X., Liu, F., Liu, P., and Yu, D. InFoBench: Evaluating Instruction Following Ability in Large Language Models. In *Findings of the Association for Computational Linguistics (ACL Findings)*, pp. 13025–13048, 2024. arXiv:2401.03601.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *Advances in Neural Information Processing Systems (NeurIPS) 36*, 2023. arXiv:2305.18290.

Razin, N., Malladi, S., Bhaskar, A., Chen, D., Arora, S., and Hanin, B. Unintentional Unalignment: Likelihood Displacement in Direct Preference Optimization. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.08847.

Rimsky, N., Gabrieli, N., Schulz, J., Tong, M., Hubinger, E., and Turner, A. 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. arXiv:2312.06681.

Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E. H., Schärli, N., and Zhou, D. Large Language Models Can Be Easily Distracted by Irrelevant Context. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pp. 31210–31227, 2023. arXiv:2302.00093.

Skalse, J., Howe, N., Krasheninnikov, D., and Krueger, D. Defining and Characterizing Reward Gaming. In *Advances in Neural Information Processing Systems (NeurIPS) 35*, pp. 9460–9471, 2022. arXiv:2209.13085.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS) 33*, pp. 3008–3021, 2020. arXiv:2009.01325.

Stolfo, A., Balachandran, V., Yousefi, S., Horvitz, E., and Nushi, B. Improving Instruction-Following in Language Models through Activation Steering. In *International Conference on Learning Representations (ICLR)*, pp. 55790–55823, 2025. arXiv:2410.12877.

Tam, Z. R., Wu, C.-K., Tsai, Y.-L., Lin, C.-Y., Lee, H.-y., and Chen, Y.-N. Let Me Speak Freely? A Study on the Impact of Format Restrictions on Large Language Model Performance. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP), Industry Track*, pp. 1218–1236, 2024. arXiv:2408.02442.

Todd, E., Li, M., Sen Sharma, A., Mueller, A., Wallace, B., and Bau, D. Function Vectors in Large Language Models. In *International Conference on Learning Representations (ICLR)*, pp. 17282–17333, 2024. arXiv:2310.15213.

Turner, A. M., Thiergart, L., Leech, G., Udell, D., Vazquez, J. J., Mini, U., and MacDiarmid, M. Steering Language Models With Activation Engineering. arXiv:2308.10248, 2023.

Wei, J., Yang, C., Song, X., Lu, Y., Hu, N., Huang, J., Tran, D., Peng, D., Liu, R., Huang, D., et al. Long-Form Factuality in Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS) 37*, pp. 80756–80827, 2024. arXiv:2403.18802.

Xu, F., Song, Y., Iyyer, M., and Choi, E. A Critical Evaluation of Evaluations for Long-form Question Answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 3225–3245, 2023. arXiv:2305.18201.

Ye, J., Wang, Y., Huang, Y., Chen, D., Zhang, Q., Moniz, N., Gao, T., Geyer, W., Huang, C., Chen, P.-Y., Chawla, N. V., and Zhang, X. Justice or Prejudice? Quantifying Biases in LLM-as-a-Judge. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.02736.

Zhao, J., Bhattacharjee, H., Korevaar, H., Radharapu, B., and El-Arini, K. Jagged Judges: Epistemic Stability Under Perturbation, Pressure, and Persistence. arXiv:2608.12645, 2026.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS) 36, Datasets and Benchmarks Track*, pp. 46595–46623, 2023. arXiv:2306.05685.

Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., Ma, X., Efrat, A., Yu, P., Yu, L., et al. LIMA: Less Is More for Alignment. In *Advances in Neural Information Processing Systems (NeurIPS) 36*, pp. 55006–55021, 2023. arXiv:2305.11206.

Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., Zhou, D., and Hou, L. Instruction-Following Evaluation for Large Language Models. arXiv:2311.07911, 2023.

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A.-K., et al. Representation Engineering: A Top-Down Approach to AI Transparency. arXiv:2310.01405, 2023.

A Appendix

This appendix gives the judge prompts and the scoring rules in full, so that the measurements in this paper can be reproduced without consulting earlier work. Every judge call below is `Claude Opus 4.8` (Anthropic, 2026), our default judge throughout the paper (Section 2).

The pairwise comprehensiveness judge. For one question we hold two answers: the unconstrained baseline and the constrained answer. The judge sees the question and both answers in the same call, and it scores each answer from 1 to 10 in terms of its comprehensiveness. Showing both answers together is what makes the omission visible, because the judge can see what one answer contains and the other leaves out. Which answer appears first is chosen at random for every pair, so the judge cannot learn a position habit. The following wording is used to ask the judge to evaluate the comprehensiveness of an answer: "How thoroughly does it cover the topic? Does it include examples, important details, edge cases, and supportive explanation?" The scale is anchored in the prompt so that the two runs stay comparable: 1–3 is very brief or shallow and misses most important aspects, 4–5 covers the basics but lacks detail, 6–7 is good coverage with reasonable depth, 8–9 is thorough and detailed with examples and nuance, and 10 could not meaningfully be improved. The judge returns only a JSON object containing the scores, which removes any need to parse prose.

The length-discounted variant. The above judge above was used for every eval. It was given no guidance about length, and we also run a second version of it as an ablation, reported in Section 2.1. It sees the same two answers, but the prompt adds four rules. Score only the distinct substantive content that addresses the question, meaning the number and importance of separate facts, steps, considerations, caveats, and concrete specifics. Ignore length, word count, verbosity, formatting, headers, bullet points, tone, polish, and writing style, and do not let any of them affect the score. Treat a shorter answer as equal to a longer one when it carries the same distinct points, and give a longer answer no credit for extra words, restatement, or generic framing. List the distinct points in each answer before scoring. The two comprehensiveness questions are also restated to ask for a count of the distinct important aspects covered rather than for thoroughness. We report this arm separately rather than folding it into the main results, because the wording is more directive and we prefer the headline numbers to come from the plainer prompt.

How the reported numbers are computed. Let b_i and c_i be the comprehensiveness scores of the baseline and the constrained answer for question i, over n questions. We report two quantities. The comprehensiveness change is the relative change of the two means,

$$\Delta = \frac{\bar{c} - \bar{b}}{\bar{b}} \times 100\%, \qquad \bar{b} = \frac{1}{n}\sum_i b_i, \quad \bar{c} = \frac{1}{n}\sum_i c_i,$$

and the unconstrained win-rate is the share of pairs in which the unconstrained answer is scored strictly higher,

$$\text{win} = \frac{1}{n}\left|\{\, i : b_i > c_i \,\}\right| \times 100\%.$$

We report both because they answer different questions. The change says how much content is lost on average, and the win-rate says how consistently it is lost, which matters when a small average could come from a few large drops. Word counts are whitespace-separated tokens of the response, and the word change is computed in the same way as Δ.

The factual-coverage judge. Comprehensiveness is a graded opinion, so we check it against a count that does not depend on the judge's taste. The procedure has two steps. First, a judge reads the unconstrained answer and breaks it into *atomic claims*: short standalone sentences, one fact or one actionable point each. The prompt tells it to drop framing sentences and transitions such as "in conclusion", to drop restatements of a claim it has already listed, and to aim for 8 to 15 claims, fewer for a shallow answer and up to 20 for a rich one. Second, for each claim in turn, a judge reads the constrained answer and answers one question: does this answer convey this claim, even in different words? The matching rules are deliberately generous about wording and strict about substance. A claim counts as covered when the answer expresses it in any phrasing, in shorter form, folded into a larger sentence, or clearly implied. It does not count when the answer omits it, contradicts it, or covers only part of it. Coverage is then the fraction of the baseline's claims that the constrained answer conveys. Because the claim list is fixed before the constrained answer is read, and because the same list is used for both answers, this measure is insensitive to length: an answer can only score well by carrying the same facts, not by being longer.

Resampling control. Two answers from the same model to the same unconstrained question already differ, so some of any measured gap is ordinary sampling noise. We therefore score a second unconstrained sample against the first with the same protocol. This gives the floor that a constrained condition has to beat before we treat it as a real loss, and all collapse numbers we report are far below it.

Which facts are kept and which are lost. The coverage numbers in Figure 4 are net figures, so they do not show whether a constraint simply cuts the answer short or changes which facts appear. Figure 6 separates the two. For each stage we take the set of pooled facts the unconstrained answer conveys and the set the constrained answers convey, counting a fact as conveyed under constraints when at least half of the eleven constrained answers convey it. The two sets are not nested. Constrained answers state facts the unconstrained answer leaves out, 254 of them after supervised fine-tuning, 116 after preference optimization and 94 after reinforcement learning, so part of the change is a reshuffle rather than pure truncation. The losses are far larger and they grow with training: 223 facts are dropped after supervised fine-tuning, against 593 and 602 after the two later stages. The supervised model is also the only one that gains about as much as it loses, which is another way of seeing that it does not collapse.

B Factual Coverage on the Benchmark

This appendix gives the per-constraint factual coverage summarized in Section 2.1, for `Qwen3-4B-Instruct` on the 160 questions and the eleven benchmark constraints (those of Figure 2 other than professional tone), 1,760 pairs with one sample each. The procedure is the factual-coverage judge of Appendix A: the claims of each unconstrained answer are extracted once (18.8 claims per answer on average) and every constrained answer

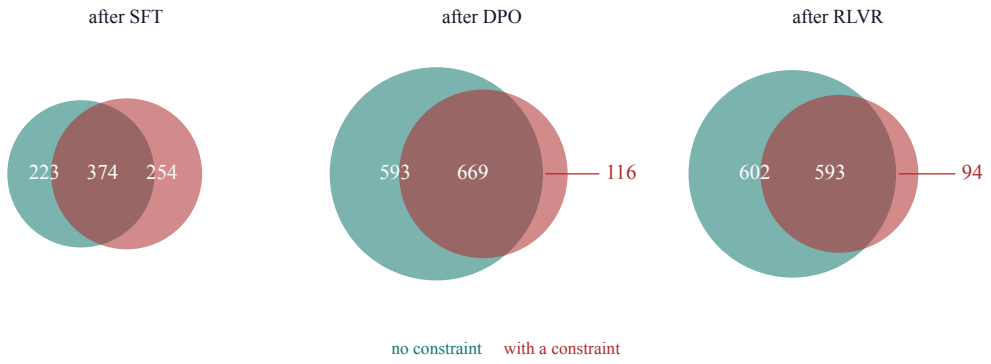

Figure 6. Facts kept, lost, and added under a constraint, at each post-training stage. Members are the pooled facts of Figure 4. The left circle is the set the unconstrained answer conveys, the right circle the set the constrained answers convey, where a fact counts as conveyed when at least half of the eleven constraints convey it. Circle areas and their overlap are proportional to the number of facts, and all three panels share one scale, so sizes are comparable across stages. Two counts sit outside their region, with a line pointing to it, because the region is too narrow to hold the label.

Table 7. Factual coverage for each constraint, `Qwen3-4B-Instruct`, 160 questions per row. *Facts kept* is the share of the unconstrained answer's claims that the constrained answer conveys, with a 95% bootstrap interval over questions; *weighted* is the same share with each claim weighted by its importance grade; *essential dropped* is the share of grade-3 claims the constrained answer omits; *length kept* is the constrained word count as a share of the unconstrained one. The last row is the resampling control: a second unconstrained answer to each question, scored against the claims of the first.

Constraint	Facts kept (%)	Weighted (%)	Essential dropped (%)	Length kept (%)
`no_mdash`	32.3 ± 2.4	37.5	45.7	34.5
`no_preamble`	31.2 ± 2.3	36.9	44.5	29.2
`no_clarifying_questions`	28.6 ± 2.4	34.0	47.4	27.9
`friendly_tone`	29.2 ± 2.3	33.7	50.6	84.0
`rendering_neutral`	31.6 ± 2.9	36.4	48.1	30.3
`mobile_display`	34.8 ± 2.4	39.8	43.8	37.9
`read_loud`	34.6 ± 2.6	40.0	43.2	40.1
`paper_print`	45.0 ± 2.7	50.4	33.4	67.0
`no_kittens`	40.6 ± 2.6	46.4	35.6	52.8
`use_word`	42.4 ± 2.7	48.0	35.6	55.9
`attn_to_form`	46.6 ± 2.8	52.1	31.1	70.8
All eleven	36.1 ± 0.8	41.4	41.7	48.2
Second unconstrained sample	58.0	—	—	100.5

is matched against that list. Two additions are made here. First, a judge grades each extracted claim once, blind to any constrained answer, from 0 (trivial) to 3 (essential to a complete answer); *importance-weighted coverage* is then the coverage with each claim weighted by its grade, and the *essential dropped* column is the share of grade-3 claims that the constrained answer fails to convey. Second, the last row of Table 7 is the resampling control of Appendix A: for every question we generate a second unconstrained answer, with a different random seed and no constraint, and score it against the claim list of the first unconstrained answer in the same way as the constrained answers. Its coverage, 58.0%, is how many of the first answer's facts a second free answer repeats, so it is the level a constrained answer would reach if the constraint cost nothing, and every constraint is read against it. Extraction, matching and grading in this appendix use `Claude Sonnet 5`. Two further results: trivial claims (grade 0) are dropped 90% of the time against 42% for essential ones, so weighting each claim by its grade raises coverage only from 36.1% to 41.4%; and the request for a friendly tone keeps 84% of the words but only 29% of the facts.

C Steering Checks

This appendix gives the details of the two checks in Section 3 that the rescue neither drops the constraint nor degrades the text.

Instruction following. `Claude Opus 4.8` reads each response on its own, with no partner text and no view of the baseline, and answers a single question: does this response obey the constraint it was given? Unconstrained prompts carry no constraint, so compliance is undefined for them. On `Qwen3-4B-Instruct`, compliance is 88.2% under the rescue against 90.5% in the collapsed, unsteered constrained condition, over 440 responses each; on `Llama-3.1-8B-Instruct` it is 97.3% against 97.0%, over 440 responses each (Table 4).

Fluency. We report the perplexity of each response under the *unsteered* model, which is the standard check that an injected direction has not pushed generation away from natural text. Scoring with the clean model matters, because the steered model's own likelihood is shifted by the vector under test, so the injected vector cannot inflate its own likelihood score. On `Qwen3-4B-Instruct`, rescued responses are about as likely as unsteered constrained ones (1.50 against 1.40 mean perplexity, over 480 responses each), and a repetition guard shows no degeneration either: distinct-3 is 0.980 against 0.988, with no repeated lines in either condition. On `Llama-3.1-8B-Instruct` the mean perplexity is 1.25 in both conditions, over 440 responses each.

D Validation of the DPO Account

The account in Section 4 says that DPO leaves the model with a single elaborated mode and no thorough alternative. We arrived at that account by inspecting the constraints that collapse, so on its own it risks being post-hoc. We check it three ways: we predict collapse in advance across a broad mix of held-out constraints; we confirm that the same mode collapses when we act on it through an aspect that has nothing to do with formatting (§D.2); and we reduce the safe side to a measurable criterion, namely whether the constraint asks for more of a feature the model already produces (§D.3).

In summary, we find that any constraint that moves the answer away from the model's default elaborated answer costs content: banning a format feature, asking for a plainer or more conversational register, or simply asking the model to be direct, which touches no formatting at all. "Give no preamble" cuts the answer by roughly two thirds and loses 36% of its content, and six further directness cues that mention no format do the same (Appendix D.2). Constraints that the default answer can absorb cost much less on the DPO checkpoint: a monospace display costs 4.4%, "your response will be printed on paper" gains 2.9%, and a no-op ban on an off-topic subject that never engages the default ("do not discuss kittens, cats or any feline animals") costs 11%, against 16 to 44% for the other eight constraints (Table 13). A constraint that asks for more of a feature the model already produces is also safe: "bold each heading" and "number each section with a Roman numeral" leave content intact (-7% each), while "italicize each heading", a feature the model uses in only 41% of its answers, costs 22% (Appendix D.3). A pre-registered out-of-sample test confirms the account (Appendix D.1): for eleven held-out constraints, predicted in advance only from whether each pushes the model off the elaborated mode, predicted and observed collapse rank-correlate at 0.78, and the one constraint predicted to be safe because it reinforces structure ("use headings") is also the one that does not collapse (-1%).

D.1 Pre-registered out-of-sample predictions

To guard against the post-hoc concern, we ran a pre-registered out-of-sample test. Before any scoring, we wrote down a prediction for eleven constraints that we had not used to build the account, and we classified each one with a single rule. A constraint should collapse the answer if it pushes the model off its elaborated mode, whether toward brevity, plainness, a casual register, or a stripped format. It should not collapse the answer if it keeps the answer inside that mode, for example by reinforcing length or structure, or if it never engages the mode at all. We then scored all eleven on `Olmo-3-7B-Instruct-DPO` with the same pairwise judge we use elsewhere, taking three samples per item against a shared unconstrained baseline, with 40 prompts each.

Table 8 reports the result, and it confirms the account. The ordering comes out right, with predicted and observed collapse rank-correlating at Spearman 0.78, and with collapse tracking the shortening that each constraint induces at $r{=}0.89$. Every constraint that we predicted would push the model off the mode does collapse. The effect ranges from mild, at -11 to -16% for the merely formal `academic` and `textbook_print`, to severe, at -68 to -75% for `eli5`, `one_sentence`, and `tweet`. The most informative case is the safe prediction, because it was the one that could have been falsified. `use_headings` reinforces structure instead of suppressing it, and it does not collapse (-1%); its constrained answer even wins 40% of the pairs. We also included two adversarial constraints as stress tests, an off-topic letter ban (`no_letter_e`) and a rigid format demand (`respond_json`). Both collapse in the same way, since both force heavy shortening (-85% and -61% of words) and lose content in proportion.

D.2 A format-free confirmation

The rewarded mode fuses length, structure, formatting, and elaboration, so a constraint can push the model off it from any of those aspects. The predictions above act on the mode mostly through format and register.

Table 8. Pre-registered out-of-sample test on `Olmo-3-7B-Instruct-DPO`. The prediction was registered before scoring. Comprehensiveness and word change are relative to the shared unconstrained baseline; pairwise judge, three samples per item, 40 prompts each.

Constraint	Predicted	Δ comp. (%)	Δ words (%)
`one_sentence`	collapse	−74	−96
`tweet`	collapse	−75	−96
`eli5`	collapse	−68	−48
`no_letter_e` (adversarial)	collapse	−64	−85
`casual_texting`	collapse	−48	−57
`respond_json` (adversarial)	collapse	−40	−61
`no_formatting`	collapse	−34	−57
`no_lists`	collapse	−25	−46
`academic`	mild	−16	−10
`textbook_print`	mild	−11	+23
`use_headings`	**safe**	**−1**	+12
Spearman(predicted, observed)			0.78
r(collapse, shortening)			0.89

Table 9. Format-free directness constraints on `Olmo-3-7B-Instruct-DPO` (same baseline and judge as Table 8). None mentions length or any format feature. The top block strips only framing; the bottom block arguably waives some content and is marked. `no_preamble` is carried over as a reference (cf. Table 13).

Constraint	Δ comp. (%)	Δ words (%)
`answer_directly`	−47	−80
`no_preamble` (reference)	−36	−70
`no_restate`	−32	−63
`no_acknowledgment`	−26	−57
`no_postamble`	−24	−53
arguably waive content (marked separately)		
`no_reasoning`	−64	−92
`no_caveats`	−38	−64
r(collapse, shortening)		0.98

To rule out the possibility that the account is secretly about formatting alone, we test the opposite aspect. We use seven constraints that grant only a *license to be direct*, and none of them mentions length, brevity, or any format feature (Table 9). Scored on the same model, with the same baseline and the same judge, all seven collapse. The reference constraint `no_preamble` gives −36%, close to its −34.5% on the same checkpoint in Table 13, which confirms the run, and collapse again tracks shortening (r=0.98). Five of the seven strip only framing, so they cannot legitimately drop any content: `no_preamble`, `no_postamble`, `no_restate`, `no_acknowledgment`, and `answer_directly`. On those five, comprehensiveness still falls by 33% on average. Even "there is no need to acknowledge the request; just respond," which removes nothing but a courtesy, costs 26%. The remaining two, `no_reasoning` and `no_caveats`, collapse further, but they arguably waive content that the user has given up, so we mark them separately. Formatting is one aspect of the mode: the model drops content whenever any aspect of the mode is disallowed.

D.3 The safe side is measurable: more of what the model already does

The account above claims that a constraint is safe when it asks for more of what the model already does, and that claim can be measured directly from the model's own output. Across the model's 440 unconstrained answers, **bold** appears in every answer (about 51 times each), and section headings and numbered sections appear about 13 times each, but *italics* appear in only 41% of answers (fewer than twice each). Bold, headings, and numbered sections are therefore features this model already uses often, and italic emphasis is not. A minimal pair isolates the effect (Table 10): "bold each section's heading" and "italicize each section's heading" are the same instruction on the same target, and they differ only in the feature requested. Bolding asks for a feature the model already uses, and it costs nothing (−7%). Italicizing asks for a feature the model rarely uses, and it collapses the answer (−22%). Adding headings or Roman-numeral section numbers, which the model also uses often, is likewise safe (−1% and −7%). This sharpens the account: what preserves content is reinforcing a

Table 10. Test of the safe side on `Olmo-3-7B-Instruct-DPO` (same baseline and judge as Table 8). Asking for more of a feature the model already produces (bold, headings, numbered sections) is safe; asking for a feature it rarely uses (italics) collapses. The last two rows are a minimal pair: the same instruction on the same target, differing only in bold vs. italic.

Constraint	Feature	Δ comp. (%)	Δ words (%)
`use_headings`	already used (headings)	−1	+12
`roman_sections`	already used (numbering)	−7	−7
bold each heading	already used (bold)	−7	−24
italicize each heading	rarely used (italics)	−22	−51

Table 11. Recovery for each constraint. Δ is the change in comprehensiveness against the unconstrained answer, and win% is the share of comparisons the unconstrained answer wins. Rows are ordered by how much the constraint costs on its own. `Qwen3-4B-Instruct`, 40 questions per cell.

	With the constraint		After "complete answer"		After "continue"	
Constraint	Δ (%)	win	Δ (%)	win	Δ (%)	win
`no_preamble`	−44.9	100	−18.7	80	−60.7	100
`no_mdash`	−44.1	100	−30.2	100	−56.6	100
`friendly_tone`	−43.1	100	−21.0	82	−59.2	100
`no_clarifying_questions`	−36.1	100	−6.1	45	−55.1	100
`rendering_neutral`	−35.5	98	+1.9	35	−50.3	98
`mobile_display`	−33.7	100	+17.8	10	−45.2	100
`no_kittens`	−31.4	100	+20.8	0	−44.3	100
`read_loud`	−22.9	98	+21.6	8	−30.2	98
`use_word`	−17.8	82	+24.8	8	−32.3	95
`paper_print`	−17.6	88	+22.4	10	−30.6	95
`attn_to_form`	−16.4	95	+23.6	8	−26.2	82
All eleven	−31.4	96	+3.6	35	−45.0	97

feature the trained model already produces. Asking the model to add a feature outside its repertoire pushes it off the mode, just as removing one of its own features does.

E Recovery, Constraint by Constraint

Section 2.1 reports that one follow-up request recovers the content. Table 11 gives the result for each constraint on its own, and adds the second follow-up we tried, a message that says only "Please continue." Its *All eleven* row is the summary reported in Section 2.1; each row is compared with the unconstrained answer to the same question, and both Δ and the win-rate are computed within those comparisons, so a win-rate near 50% means the two answers are about equal.

That second message does not recover the content. It leaves the answer further below the unconstrained one than the constrained answer already was, on every constraint. This is the reason we read the recovery as an effect of asking for a complete answer rather than an effect of taking one more turn. We note one limit of the comparison: an answer that continues an earlier answer is only a part of an answer, and we score that part on its own, so some of its low score follows from the form of the request and not from the model withholding anything. The request for a complete answer does not have this problem, because the model answers again from the start.

F Constraint Texts

Every condition in this paper is one sentence, or a short group of sentences, appended to the question. Table 12 gives all of them word for word, grouped by the experiment that uses them, so that the conditions can be reproduced exactly. The identifiers match our released code and result files. A constraint that appears in more than one group is written out once.

Table 12. Verbatim text of every constraint used in the paper.

Identifier	Text appended to the question
Deployment panel *Figure 2, Tables 1 and 5*	
`no_mdash`	"Do not use em dashes (—) or en dashes (–) anywhere in your response. Use commas, semicolons, parentheses, or periods instead."
`friendly_tone`	"Use a warm, friendly, and conversational tone. Use contractions (such as 'don't,' 'you'll,' 'it's') and approachable language. Speak as if you're chatting with a friend."
`no_clarifying_questions`	"Do not ask the user any clarifying questions. Provide your best answer using only the information given in the prompt."
`read_loud`	"Your response will be read aloud by a text-to-speech system."
`no_preamble`	"Do not begin your response with a greeting, acknowledgment, or conversational opener such as 'Certainly!', 'Great question!', 'I'd be happy to help!', 'Sure!', or 'Of course!'. Start directly with the first substantive sentence of your answer."
`mobile_display`	"Your response will appear in a chat window on a mobile device."
`paper_print`	"Your response will be printed on paper."
`no_kittens`	"Do not discuss kittens, cats or any feline animals in your response."
`attn_to_form`	"Pay attention to your formatting and punctuation as you write."
`rendering_neutral`	"Your response will be displayed in a monospace font."
`use_word`	"Freely use any words in your response."
Register and hedging set *Section 4;* `professional_tone` *also appears in Figure 2*	
`professional_tone`	"Do not use exclamation marks, casual language, or informal expressions. Maintain a strictly professional and formal tone throughout your response."
`hedging`	"Avoid making definitive or absolute claims. Use hedging language such as 'may,' 'might,' 'could,' or 'evidence suggests' instead of stating facts directly."
`plain_language`	"Write at a reading level accessible to a general audience. Avoid all technical jargon, acronyms, and complex sentence structures. Use simple, everyday words and short sentences."
`no_preamble`	*as above*
Held-out set *pre-registered test of Appendix D*	
`one_sentence`	"Answer in a single sentence."
`tweet`	"Keep your entire answer under 280 characters."
`eli5`	"Explain it so a five-year-old could understand."
`casual_texting`	"Reply in a casual style, the way you would text a friend."
`no_formatting`	"Respond in plain text only. Do not use markdown, bullet points, numbered lists, headings, or bold."
`no_lists`	"Do not use any bulleted or numbered lists."
`academic`	"Write in a formal academic style suitable for a journal."
`textbook_print`	"Your response will be printed in a reference textbook."
`use_headings`	"Organize your answer with clear section headings."
`no_letter_e`	"Do not use the letter 'e' anywhere in your response."
`respond_json`	"Respond only as a JSON object."

Table 12, continued.

Identifier	Text appended to the question
	“Be thorough and include all relevant details.”
`be_thorough`	
	“Address the question in the best possible way.”
`best_way`	
Directness cues *Appendix D.2*	
	as above, carried over as the reference condition
`no_preamble`	
	“Do not end your response with a summary, conclusion, wrap-up, or closing remark such as ‘In summary’, ‘To conclude’, or ‘I hope this helps’. Stop once you have given the answer.”
`no_postamble`	
	“Do not restate or rephrase my question before answering.”
`no_restate`	
	“Answer directly and get straight to the point.”
`answer_directly`	
	“Do not explain your reasoning; give only your conclusion.”
`no_reasoning`	
	“Skip any caveats, disclaimers, or qualifications.”
`no_caveats`	
	“There is no need to acknowledge the request; just respond.”
`no_acknowledgment`	
Structure variants *Appendix D.3*	
	“Start each section with a Roman numeral (I, II, III, and so on).”
`roman_sections`	
	“Do not capitalize the first letter of any new section.”
`lowercase_sections`	
	“Start each section with the current date and time.”
`timestamp_sections`	
	“Do not include any math or equations in your response.”
`no_math`	
	“Make each section exactly five sentences long.”
`five_sentence_sections`	
	“Every section must have exactly three bullet points.”
`three_bullets_sections`	
	“Italicize three random words in each section.”
`italic_random`	
	“Bold each section’s heading.”
`bold_heading`	
	“Italicize each section’s heading.”
`italic_heading`	

G THE RLVR INSTRUCTION-FOLLOWING REWARD

Section 4 says that the reward used in the reinforcement-learning stage pays nothing for content. This appendix shows the reward itself and works through one example, so the claim can be checked rather than taken on trust.

The reward. `OLMo`’s instruction-following reward is computed by the `IFEvalVerifier` in the public `open-instruct` code. A training prompt carries a list of K constraint identifiers, each naming a small program that reads the answer and returns true or false. Writing y for the answer and check_k for the k-th program, the reward is the fraction of the checks that pass:

$$R(y) \;=\; \frac{1}{K}\sum_{k=1}^{K} \mathbb{1}[\mathrm{check}_k(y)]\,.$$

The only other case in the code is an empty answer, which scores 0. There is no term for length, for coverage, for accuracy, or for whether the question was answered at all. Every program is a form priming test that can be settled by scanning the text, for example `punctuation:no_comma`, `detectable_format:number_bullet_lists`, `change_case:english_lowercase`, or `keywords:forbidden_words`.

What this means for a thorough answer. Because R counts only satisfied checks, two answers that satisfy the same checks receive the same reward, however much they differ in substance. Consider a training

Table 13. Per-constraint collapse on either side of the reinforcement-learning stage, on the released `Olmo-3` checkpoints. Δ is the change in comprehensiveness against the same checkpoint's own unconstrained answer, over 40 questions per cell with one sample per pair, scored by the same pairwise judge in both columns. Rows are ordered by how much preference optimization alone cost, which makes the flatness of the last column visible: the additional cost does not depend on how damaging the constraint already was.

Constraint	After DPO (%)	After RLVR (%)	Additional (%)
`paper_print`	+2.9	−6.0	−8.9
`rendering_neutral`	−4.4	−16.2	−11.8
`no_kittens`	−11.3	−21.3	−10.0
`read_loud`	−15.6	−24.1	−8.5
`attn_to_form`	−16.7	−22.2	−5.5
`use_word`	−25.3	−37.3	−12.0
`mobile_display`	−30.4	−34.3	−3.9
`no_preamble`	−34.5	−45.4	−10.9
`no_mdash`	−35.7	−49.1	−13.4
`no_clarifying_questions`	−36.6	−44.1	−7.5
`friendly_tone`	−44.2	−49.0	−4.8
All eleven	−22.9	−31.7	−8.8
SD across constraints	15.0	14.6	3.2

prompt that asks a real question and attaches two constraints, one forbidding commas and one requiring the answer in lowercase:

> *Prompt.* Explain how a neural network learns through backpropagation. Do not use any commas in your response. Your entire response should be in lowercase.

A thorough answer that walks through the forward pass, the loss, the backward pass, and the weight update, and that obeys both constraints, satisfies two checks out of two and scores $R = 1.0$. The answer "backpropagation adjusts weights using gradients" also satisfies two checks out of two and also scores $R = 1.0$. The reward cannot separate them. Worse, the short answer earns that score more reliably, because a long answer has more places in which a stray comma or a capital letter can appear, and one such slip halves the reward to 0.5. Optimizing R therefore pushes the policy toward the shortest answer that passes, which is what we observe at the end of training.

Why this should not happen for math or code. The same recipe uses a different verifier for those domains, one that checks the final answer or runs the tests. There the thing being checked is the content the user wants, so an answer cannot score well while leaving the substance out. The problem is specific to rewards that only inspect the form of the response.

H The Reinforcement-Learning Stage Applies a Constant Offset

Section 4 claims that the reinforcement-learning stage costs the same amount under every constraint. This appendix gives the per-constraint measurement behind that claim.

We score the two released checkpoints from either side of the stage, `Olmo-3-7B-Instruct-DPO` after preference optimization and `Olmo-3-7B-Instruct` after reinforcement learning, on the eleven deployment constraints with 40 questions each and one sample per pair. Both columns use the same pairwise judge, so the comparison between them is internal and does not rely on matching any earlier run.

Every one of the eleven constraints costs more after the stage. The additional cost is close to constant: its mean is -8.8 points, its standard deviation is 3.2, and it ranges from -3.9 to -13.4. Two functional forms can be fitted to the pair of columns. An additive model, in which the stage subtracts a fixed amount under every constraint, leaves a residual standard deviation of 3.2. A multiplicative model, in which the stage amplifies whatever preference optimization already did, leaves 6.3. The free fit prefers the additive form, returning a slope of 0.96 whose confidence interval contains one, and an intercept of -9.8. The 95% confidence interval of the slope runs from 0.80 to 1.11. The spread across constraints separates the two readings as well: had the stage merely pushed the preference-optimization mechanism further, that mechanism's selective damage would have scaled and the spread would have widened, but it is unchanged (15.0 before against 14.6 after, Table 13).

The slope is measured reliably here, which is worth stating because a regression of one noisy stage on another is not always trustworthy. Attenuation depends on the ratio of judge variance to the spread being regressed.

That spread is 15 points across constraints, against roughly two points of judge variance for this panel, so the attenuation factor is 0.98 and the estimate is essentially unbiased. We report the slope rather than a correlation between the additional cost and the earlier cost, because the earlier cost would then appear on both sides and produce a negative correlation from measurement noise alone.

I A Fix at the Source

The two mechanisms of Section 4 share a single fix: make retained content the quantity that the reward optimizes. In practice, the reward should score how much of a comprehensive reference answer's substance the response conveys, measured relative to that reference and blind to length; it should treat constraint satisfaction as a floor that has to be met rather than a term that can be traded against content; and it should penalize fabrication. Such a reward addresses both mechanisms at once. It removes RLVR's incentive to give the minimal compliant response, because a short answer no longer earns the full reward. It also detaches content from the formatted style that DPO rewards, because substance is scored directly instead of through length or structure. Two checks that need no training already pass. An additive reward with a satisfaction floor keeps a usable learning signal, where a multiplicative form does not, and on real data the reward ranks thorough answers above collapsed ones and collapsed ones above one-line answers. The decisive test is a short reinforcement-learning run with this reward in place, and we leave that to future work.

J The Same Trigger on a Grounded Task

Section 4 says that a grounded reward protects content. This appendix runs the deployment constraints on a reasoning model whose training reward checks the final answer, and asks two questions. Does the form cue still change generation, and does the graded content survive.

Setup. The model is `Qwen3-4B-Thinking-2507`. The problems are the 30 problems of AIME 2026, which were released after the model's training cutoff, with 8 samples per problem and a budget of 81,920 tokens, of which 0.2% of rollouts are truncated. Each constraint is appended to the problem word for word, and each constrained rollout shares its sampling seed with an unconstrained rollout of the same problem, so the two differ only in the constraint text. We measure the length of the reasoning trace, in tokens between the thinking delimiters, and the accuracy of the boxed final answer, checked with `math-verify`. Each effect is the paired per-problem change against the unconstrained rollouts of the same problem, averaged over problems, with a 95% cluster bootstrap over problems (2,000 resamples). Two constraints, `plain_language` and `hedging`, ask the model to avoid the definitive, formatted answer that the grader reads, so a missing boxed answer would count as wrong; in the run this happens in one rollout of 240 for each, and we report their accuracy with that caveat. We also ran MATH-500, but it predates the model's training data, so we do not report it.

Result. Table 14 gives the numbers. The tone constraints shorten the trace, and the intervals exclude zero: plain language by 13.7%, a friendly tone by 9.7%, and a professional tone by 5.4%. The no-op constraint sits at -1.8% with an interval that spans zero, and the two restrictions, `no_preamble` and `no_mdash`, are inside it. Accuracy is 81.2% without a constraint. Every constrained arm is within 4 points of that, every interval includes zero, and the widest interval reaches -9.6 points, so the data rules out a cost above about 10 points and does not rule out a cost of a few points.

Two things follow. The constraint reaches the model on a grounded task as it does on an open-ended one, since it changes how long the model thinks. And the content that the reward checks survives, within the precision of 30 problems. Trace length is not comprehensiveness: a shorter trace with the same answer is not a loss the user can see, so we do not call this a collapse. What separates the two settings is what the reward protects. We test one model and one domain, and do not test code.

K Extended Related Work

This appendix gives the full discussion that Section 1.1 summarizes.

Constraint following and its costs. Instruction-following benchmarks score whether a stated constraint is satisfied (Zhou et al., 2023b; Jiang et al., 2024; Qin et al., 2024; Pyatkin et al., 2025), not what obeying it costs, so a terse compliant answer scores as well as a thorough one. Baghaei Potraghloo et al. (2026) measured that cost for lexical bans; we show that constraints which ban nothing an answer would have used, or nothing at all, still cost 15 to 38% of the content (Table 2). Stolfo et al. (2025) also report a quality cost from adding a constraint, measured with a single-response rubric of the kind we show undercounts the loss (Section 2.1). Format restrictions and irrelevant clauses are known to lower accuracy on tasks with a right answer (Tam et al., 2024; Shi et al., 2023; Mirzadeh et al., 2025), and hard constrained decoding can exclude the intermediate

Table 14. Constraints on a reasoning model, `Qwen3-4B-Thinking-2507`, AIME 2026, 30 problems, 8 samples each. Δ trace is the paired per-problem change in reasoning-trace tokens against the unconstrained rollouts, and Δ acc. the paired change in accuracy, both with 95% cluster bootstrap intervals over problems. `plain_language` and `hedging` ask the model to avoid the definitive, formatted answer the grader reads; one rollout in 240 of each has no boxed answer and is counted as wrong. `use_word` is the no-op constraint of Table 2.

Constraint	Trace (tok)	Δ trace (%) [95% CI]	Acc. (%)	Δ acc. (pp) [95% CI]
Unconstrained	20,935	—	81.2	—
`plain_language`	17,715	−13.7 [−18.9, −7.3]	78.8	−2.5 [−7.1, +1.2]
`friendly_tone`	18,973	−9.7 [−13.4, −5.9]	77.9	−3.3 [−7.1, +0.0]
`professional_tone`	19,825	−5.4 [−9.3, −1.5]	77.5	−3.8 [−9.6, +1.2]
`no_preamble`	20,429	−2.6 [−5.7, +0.5]	79.2	−2.1 [−6.2, +1.7]
`no_mdash`	20,634	−1.8 [−6.2, +3.3]	79.2	−2.1 [−5.8, +0.8]
`hedging`	20,419	+1.4 [−4.1, +7.5]	77.9	−3.3 [−7.1, +0.0]
`use_word`	20,430	−1.8 [−5.6, +3.2]	80.0	−1.2 [−5.0, +2.9]

tokens a derivation needs (Banerjee et al., 2025). Our constraints don't restrict the token set nor the grammar, and we measure what an open-ended answer leaves out, which an accuracy check cannot see.

Judge protocols and measuring omission. Comprehensiveness is a recognized dimension of long-form answer quality that automatic metrics capture poorly (Xu et al., 2023). Length and style biases are a known confound in LLM-as-a-judge scoring (Hu et al., 2025), and judges can also disagree with each other under adversarial rewording or repeated querying (Ye et al., 2025; Zhao et al., 2026). We adopt the side-by-side protocol of Zheng et al. (2023). Judges of this kind reward length and style (Dubois et al., 2024; Feuer et al., 2025), so we also report a length-discounted variant (Table 3), and we build on atomic-fact evaluation, which scores a response claim by claim (Min et al., 2023), in the spirit of long-form factuality decomposition (Wei et al., 2024). What we change is the reference. Prior protocols score a response against an external source or a competing model; we score it against the answer the same model gives without the constraint, so a response loses ground only by dropping content the model itself supplied. Scored this way, pointwise protocols report about half the drop that a side-by-side comparison finds (Section 2.1).

Locating behaviors in activations. A single direction in the residual stream can carry a behavior, and adding or subtracting it controls that behavior, as shown for sentiment (Turner et al., 2023) and refusal (Arditi et al., 2024), generalized to concepts read from contrastive prompt pairs (Zou et al., 2023; Rimsky et al., 2024), and related to function-vector representations of task behavior more broadly (Todd et al., 2024). Stolfo et al. (2025) compute instruction vectors as we compute our collapse direction, from the difference of mean activations with and without an instruction, and steer with them to improve adherence, adding them for most constraints and subtracting word-inclusion vectors to enforce exclusion. With a pointwise judge, they also find that constraints lower response quality, and they treat this as a trade-off between adherence and comprehensiveness, since satisfying a constraint "may lead to less comprehensive responses". We show that pointwise judges like theirs capture only about half of the loss that a pairwise judge detects (Section 2.1). More importantly, we demonstrate that the trade-off is not inherent: no-op constraints that require nothing still cost content, a follow-up request recovers it with the constraint in force, and subtracting our direction while the constraint stays in the prompt restores most of it, with compliance within about two points of the unsteered answers. Finally, we trace the collapse's root cause, which they do not examine, to reward-based post-training (Section 4).

Post-training rewards and their blind spots. Optimizing a proxy reward too hard moves a model away from the behavior the proxy stands for (Gao et al., 2023), a special case of the broader reward misspecification and reward gaming problem (Pan et al., 2022; Skalse et al., 2022). The pipeline we study (Team Olmo, 2025) optimizes two proxies that do not measure content: preference optimization (Rafailov et al., 2023) rewards whatever separates preferred from dispreferred answers, in practice length and formatting, and the RLVR instruction-following verifier (Lambert et al., 2025) pays only for constraint satisfaction. Two known failures of preference optimization could be mistaken for our effect: a preference for length (Stiennon et al., 2020; Park et al., 2024; Lu et al., 2024), which SimPO (Meng et al., 2024) was built to curb, and likelihood displacement (Razin et al., 2025). Ours is neither. It survives when the two answers are the same length (Section 2.1), standard and length-normalized objectives lose the same content at matched quality (Table 6), and it is conditional on the constraint being present rather than a property of the trained distribution (Table 2). Post-training changes a model's style more than its knowledge and narrows the outputs it produces (Zhou et al., 2023a; Lin et al., 2024; Kirk et al., 2024), which is what ties thoroughness to a single elaborated answer (Section 4).