Format Collapse in Policy Optimization for LLM Reasoning

Anonymous ACL submission

Abstract

Policy optimization (PO) is widely used to post-train language models for reasoning, often with a rigid <think><answer> format. We study what PO learns when this constraint is relaxed. We evaluate a relaxed reward across six models and four benchmarks, then audit learned formats on GSM8K and controlled variants across three on-policy algorithms. In these audits, accepting either explicit reasoning or a direct answer pushes the policy toward the shortest valid format: an <answer>-only response. On GSM8K, the collapsed Qwen-2.5 7B policy produces answer-only outputs in 98.6% of generations and loses 7.1 pass@5 points relative to a strict-format baseline. Controlled rewards show this is not a regex artifact: with nested formats, PO learns easy requirements before harder ones despite a 4× hard-format reward; with mutually exclusive equal-reward formats, the audited Qwen-2.5 7B final policy assigns more than 99% of generations to the easiest format. A sample-gated gradient account fits the pattern: formats receive positive updates only when sampled, so early support can outweigh reward scale. KL-controlled runs show collapse appears only when the policy can move far enough from initialization. The result identifies a practical failure mode for RLVR reward design.

1 Introduction

Reinforcement Learning with Verifiable Rewards (RLVR) (Zhu et al., 2025; Wang et al., 2025; Wen et al., 2025) is a common way to improve language-model reasoning (Yu et al., 2025; Qwen et al., 2025; Grattafiori et al., 2024). Most successful recipes make the final answer verifiable and also constrain the output format, typically by requiring a <think> block followed by an <answer> block (DeepSeek-AI et al., 2025; Liu et al., 2025a; Xie et al., 2025; Chen et al., 2025). This design works, but it also hides a basic question about the optimizer: when several output formats all satisfy the reward, which one does on-policy optimization learn?

We answer this question by giving the policy a controlled choice between a structured response with explicit reasoning and a direct <answer>-only response. The reward accepts both. We evaluate strict-vs-composite rewards across six models from five families and four benchmarks spanning math, code, logic, and planning, then audit GSM8K output distributions and controlled-format variants across three policy optimization (PO) algorithms (GRPO (Shao et al., 2024), DAPO (Yu et al., 2025), REINFORCE++ (Hu et al., 2025)). Accuracy results usually favor strict formatting. On GSM8K, output-distribution audits show that the trained policy does not choose formats adaptively: it shifts to the shortest accepted format and usually omits explicit reasoning. We call this behavior **format collapse**.

The collapse is not only a formatting curiosity. In the composite-reward setting, the Qwen-2.5 7B policy emits the <answer>-only format in 98.6% of generations on GSM8K and loses 7.1 pass@5 points relative to the strict-format baseline. Across the six tested models, more than 95% of generations use the answer-only shortcut. This links the format shift to the accuracy drop rather than treating Table 1 as a stand-alone comparison.

We then separate this observation from the particular composite reward. First, we build a nested hierarchy of format requirements: $\mathcal{F}_3 \subset \mathcal{F}_2 \subset \mathcal{F}_1$, so every response satisfying the hardest format also satisfies the medium and easy requirements. Even though the maximum-reward response satisfies all three, the optimizer learns them in order: easy, then medium, then hard. Second, we make the hardest format worth 4× the easiest, yet the same early ordering remains. Third, we remove shared structure by using mutually exclusive format languages with equal reward; in our audited Qwen-2.5 7B

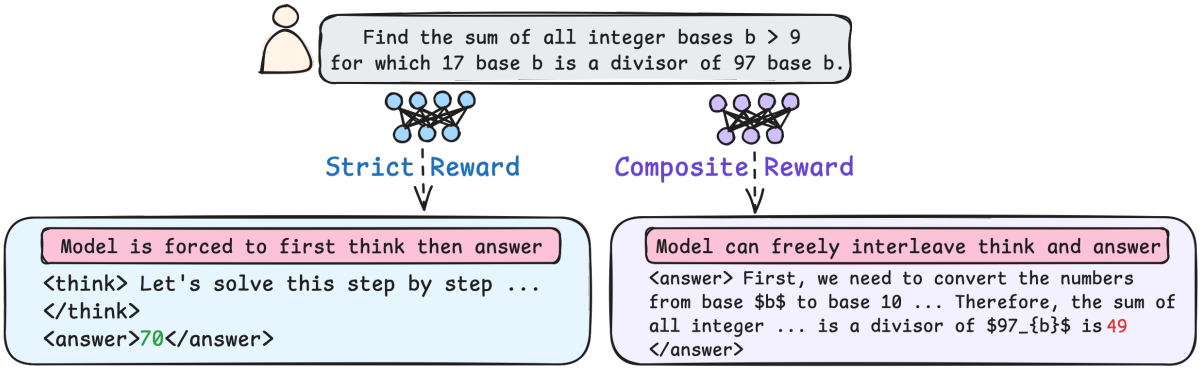

Figure 1: We compare a strict reward that accepts only a *think-then-answer* response with a composite reward that accepts both explicit reasoning and direct-answer formats. Under the composite reward, policy optimization consistently shifts probability mass to the shortest valid format.

run, the final policy still assigns more than 99% of generations to the easiest format.

The mechanism is sample-gated. In on-policy RL, a format reward affects the update only on completions that already satisfy that format. A rarely sampled high-reward format can therefore receive less early signal than a lower-reward format that the current policy samples often. In expectation, the reward-weighted signal scales as $r_i p_i(\theta)$, so a high-reward format can still receive less early signal when current-policy support is low.

Finally, we study the role of exploration. Recent RLVR methods often reduce or remove KL penalties to let policies move farther from the initial model (Yu et al., 2025; Yue et al., 2025). In our controlled runs, a low KL penalty permits format collapse, while a higher penalty prevents the same shortcut from emerging. This does not prove a universal convergence law, but it shows that the collapse is an active policy shift enabled by exploration, not passive retention of the base model's behavior.

Our contributions are as follows:

❶ We identify **format collapse** in RLVR: when multiple valid response formats are rewarded, PO converges to the shortest accepted format, often at a measurable cost to reasoning accuracy.

❷ We operationalize format difficulty by learnability and show a consistent least-resistance ordering across nested, weighted, and mutually exclusive reward designs.

❸ We evaluate strict-vs-composite rewards across six models and four benchmarks, audit GSM8K output distributions, and reproduce the least-resistance pattern across three on-policy algorithms.

❹ We give a sample-gated policy-gradient account of the effect and show, through KL-controlled runs, that collapse requires enough policy movement from initialization.

2 Related Work

RLVR for reasoning. Recent reasoning systems use verifiable rewards to improve final-answer accuracy, often together with a prescribed reasoning format (DeepSeek-AI et al., 2025; Shao et al., 2024; Yu et al., 2025; Liu et al., 2025a; Chen et al., 2025; Xie et al., 2025). Our work studies a smaller but consequential design choice in these pipelines: what happens when the reward accepts more than one syntactically valid path. We do not propose a new optimizer. Instead, we use existing on-policy algorithms to isolate a reward-specification failure mode that can appear even when correctness remains verifiable.

Shortcut learning and sparse rewards. Shortcut learning is usually discussed as a response to dataset bias, environment structure, or objective misspecification (Sohn et al., 2021; Kurita and Søgaard, 2019; Jin et al., 2020; Liu et al., 2022; Mead et al., 2025). Sparse-reward RL also has a long history of agents learning the first accessible reward channel rather than the designer's intended behavior. Our setting differs in two ways. First, the shortcut is an output-language choice inside RLVR for language models, not a physical trajectory or offline-data artifact. Second, we can define the competing paths exactly with regular languages and then test nested, weighted, and mutually exclusive variants. This lets us distinguish shared-substructure effects from a direct choice between disjoint formats.

Positioning. The closest concern is that learning easier objectives before harder ones may be unsurprising. We agree, and we frame our claim accordingly: the contribution is not that easy com-

ponents can be learned first, but that this ordering dominates practical RLVR format rewards strongly enough to erase explicit reasoning, reduce task accuracy, and persist under a 4× counter-incentive. The result is therefore a warning about reward design rather than a claim of a new universal law of optimization.

3 Setup

We evaluate flexible format rewards across models, tasks, and on-policy optimizers. All reported task results use pass@5 accuracy, and all format analyses classify generated strings with the same regular expressions used for reward computation.

3.1 Models

We use six public models from five families: Gemma-3 4B and 12B (Team et al., 2025), Qwen-2.5 7B (Qwen et al., 2025), Llama-3.1 8B (Grattafiori et al., 2024), Ministral 8B (Jiang et al., 2023), and Yi 6B (AI et al., 2025). The suite is intentionally modest in scale but diverse enough to check whether the effect is tied to one model family. Experiments used a private GPU cluster; further training details are in Appendix B.1.2.

3.2 Datasets

The tasks cover math, code, logic, and planning: GSM8K (Cobbe et al., 2021), Math-Hard (Hendrycks et al., 2021), Open-R1 Math 220k (Hugging Face, 2025), rStar-Coder (Liu et al., 2025b), Open-R1 verifiable coding (Hugging Face, 2025), Knights and Knaves (Xie et al., 2024), ReClor (Yu et al., 2020), and Planning-Mystery for planning. Table 1 reports the four benchmarks used for the main strict-vs-composite comparison; the controlled reward studies use the same prompting and parsing machinery.

3.3 Policy Optimization Algorithms

We test three on-policy algorithms used in recent RLVR work: GRPO (Shao et al., 2024), DAPO (Yu et al., 2025), and REINFORCE++ (Hu et al., 2025). GRPO and DAPO use group-relative advantages and clipped policy updates, with Dr. GRPO corrections for response length and question difficulty (Liu et al., 2025c); REINFORCE++ provides an actor-only policy-gradient baseline. Table 2 gives the hyperparameters.

We used LoRA adapters for all training runs and targeted all linear projection layers. For REINFORCE++, which requires a reference model, the policy and reference used separate LoRA adapter heads on the same base model. A typical run required about 8.5 GPU hours, and the full experimental suite required approximately 200 GPU hours. Appendix B.1.2 gives additional compute details.

3.4 Operational Definitions

We use *format difficulty* operationally. For a format language $\mathcal{F}$ and threshold τ, define $K_\tau(\mathcal{F})$ as the first optimization step at which the policy's measured compliance with $\mathcal{F}$ exceeds τ under the same optimizer, data, and prompt. We use this statistic only to compare formats inside a controlled run, not as a property of a regex alone. In our experiments, the ordering induced by K_τ matches the number of required structural constraints: a boxed answer is easiest, `<answer>` plus boxed answer is harder, and `<think><answer>` is hardest. Two format rewards are *disjoint* when their regex languages have empty pairwise intersection, so a generation can receive at most one of the corresponding rewards. Appendix B.1.3 lists the exact regular expressions.

For the mutually exclusive condition, the disjointness check is part of the main setup. The easy language ends immediately after a boxed answer, the medium language must be an entire `<answer>` response containing a boxed answer, and the hard language must be an entire `<think><answer>` response containing a boxed answer. Start- and end-of-string anchors make these languages pairwise disjoint; Appendix B.1.3 gives the exact regular expressions.

3.5 Prompt Controls and Output Classification

Prompt wording can itself bias the policy toward or away from explicit reasoning. We therefore separate the prompt from the reward language. For the composite and hierarchy experiments, the prompt permits thinking, answering, and interleaving the two, but it does not rank the valid formats. For the strict baseline, the prompt requires the usual think-then-answer order. Appendix B.2 gives the full text.

This control is important for interpretation. The composite prompt does not instruct the model to prefer answer-only outputs; it merely leaves multiple syntactic paths available. When a composite-trained policy collapses to the direct-answer format, the shift is therefore attributed to the learned

Table 1: Pass@5 accuracy (%) after training with a **Strict Reward**, which accepts only `<think><answer>`, or a **Composite Reward**, which also accepts direct `<answer>`-only responses. Strict formatting usually improves accuracy, but Section 4.2 shows that the key effect is the learned output distribution, not the aggregate accuracy alone. Best result in each pair is bolded.

	GSM8K		rStar-Coder		ReClor		Planning-Mystery	
Model	**Strict**	**Composite**	**Strict**	**Composite**	**Strict**	**Composite**	**Strict**	**Composite**
Gemma-3 4B	72.4	**73.1**	**55.8**	53.5	**73.2**	69.4	**58.3**	55.9
Qwen-2.5 7B	**92.4**	85.5	73.2	**73.6**	**80.1**	73.8	**77.5**	73.8
Llama-3.1 8B	**86.2**	82.9	**64.1**	59.2	**78.3**	72.7	**74.9**	70.6
Ministral 8B	**89.5**	84.8	**74.0**	72.6	**81.0**	76.5	76.8	**77.3**
Yi 6B	**85.1**	83.8	**68.3**	63.7	**68.4**	65.1	**71.2**	67.7
Gemma-3 12B	**94.6**	86.4	**76.1**	72.3	**85.5**	81.2	**81.2**	76.5

Table 2: Training configuration used to keep reward comparisons controlled. Shared optimizer, batching, and LoRA settings are separated from the few algorithm-specific settings, so differences among the strict, composite, nested, weighted, and mutually exclusive results come from reward language rather than per-condition hyperparameter tuning.

Shared across algorithms

Shared setting	Value
Adam β_1/β_2	0.9 / 0.99
Weight decay / grad norm	0.1 / 0.15
LR schedule / warmup	cosine / 0.1
Batch size / epochs	8 / 3
Precision	bfloat16
LoRA rank / alpha	64 / 128
LoRA targets	all linear projections

Algorithm-specific settings

Algorithm setting	GRPO	DAPO	REINFORCE++
Learning rate	5×10^{-6}	5×10^{-6}	5×10^{-5}
Loss type	*dr_grpo*	*dr_grpo*	N/A
ϵ / ϵ_{high}	0.2 / 0.2	0.2 / 0.28	N/A
KL β	0.1 / 0.3	0.1 / 0.3	N/A
Generations / prompt	5	5	1
PPO epochs	N/A	N/A	2
Adv. / reward norm.	N/A	N/A	True / True

reward path rather than to a prompt that names the direct-answer format as preferred.

For format-distribution analyses, we sample from the trained policy and then classify outputs with the same regular expressions used for reward computation. The classification is independent of correctness: a generation is counted as answer-only, think-then-answer, or other based only on its syntax. For the GSM8K format-collapse analysis, we generated 600 samples from the test set for each trained model before computing the distributions in Table 4.

4 Identifying Format Collapse

We begin by relaxing the rigid format constraints used in many RLVR pipelines. The experiment asks a concrete question: when two response formats both receive the same format reward, which one does the optimizer learn?

4.1 Defining a Choice via Reward Functions

Let y be a model-generated sequence. In all reasoning experiments, the training reward is the sum of a correctness reward and a format reward. The correctness reward checks whether the extracted final answer is equivalent to the gold answer; the format reward is one of the following.

The **Strict Reward**, $R_{\text{strict}}(y)$, is positive only if y follows a fixed `<think><answer>` structure.

$$R_{\text{strict}}(y) = \begin{cases} +3 & \text{if } y \text{ matches the strict format} \\ -4 & \text{otherwise} \end{cases} \tag{1}$$

The **Composite Reward**, $R_{\text{composite}}(y)$, accepts either interleaved `<think>` and `<answer>` blocks or a direct `<answer>`-only response. Let $\mathcal{F}_{\text{comp}}$ denote this union of valid formats.

$$R_{\text{composite}}(y) = \begin{cases} +3 & \text{if } y \in \mathcal{F}_{\text{comp}} \\ -4 & \text{otherwise} \end{cases} \tag{2}$$

Appendix B.1.3 gives the regular expressions, and Appendix B.2 gives the prompts.

If the policy used the relaxed reward adaptively, it could reserve explicit reasoning for harder examples. We instead observe the opposite.

4.2 Convergence to a Minimal Format

Table 1 compares strict and composite rewards across tasks. A detailed version is in Table 7. The

Table 3: Reward-language contract for the controlled experiments. The key distinction is whether valid formats are alternatives, nested components, or mutually exclusive choices, since this determines whether the optimizer can collect several format rewards or must choose exactly one. Appendix B.1.3 gives the exact regular expressions.

Setting	**Accepted format language**	**Format reward aggregation**
Strict	Only a full think-then-answer response with separate thinking and answer blocks.	Constant reward if the single strict language matches.
Composite	Either the strict/interleaved reasoning format or a direct answer-only response.	Constant reward if any accepted language matches.
Nested hierarchy	Three nested languages: boxed answer $\mathcal{F}_1$, answer-tagged boxed answer $\mathcal{F}_2$, and full think-then-answer $\mathcal{F}_3$, with $\mathcal{F}_3 \subset \mathcal{F}_2 \subset \mathcal{F}_1$.	Sum of rewards for all satisfied languages, so matching $\mathcal{F}_3$ also earns the $\mathcal{F}_2$ and $\mathcal{F}_1$ rewards.
Weighted hierarchy	Same nested languages as the hierarchy setting.	Same additive aggregation, with rewards scaled as 1:2:4 for $\mathcal{F}_1$:$\mathcal{F}_2$:$\mathcal{F}_3$.
Mutually exclusive	Three anchored languages: boxed-answer-only, answer-tag-only, and think-then-answer. The languages are pairwise disjoint.	Constant reward for the single matched language; a generation can match at most one language.

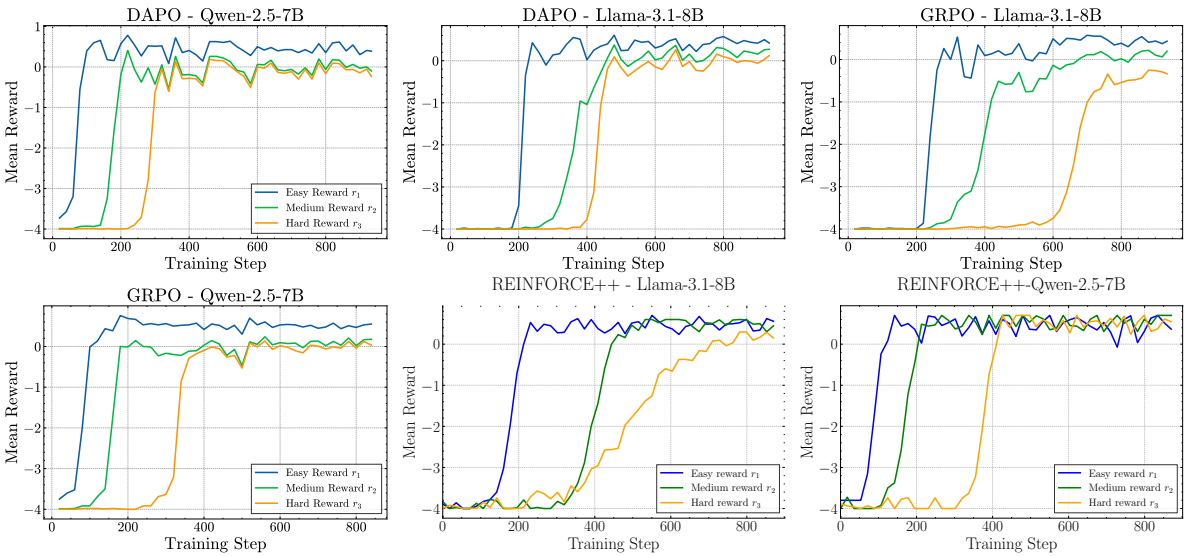

Figure 2: Nested reward hierarchy. Each line is a format reward measured during training. Across model-algorithm pairs, the easiest requirement, f_1, rises before f_2, and the full `<think><answer>` format, f_3, rises last.

strict reward wins most pairs, but the more important evidence is the format distribution learned under the composite reward.

On GSM8K, the strict reward outperforms the composite reward for every model, with gaps of **7.1** points for Qwen-2.5 7B (92.4 vs. 85.5) and **8.2** points for Gemma-3 12B (94.6 vs. 86.4). Similar gaps appear on rStar-Coder and ReClor.

The trained composite policies overwhelmingly choose the shortest valid format. Across all six models on GSM8K, at least 95.7% of generations are `<answer>`-only, and fewer than 4% contain the full `<think><answer>` structure (Table 4). We refer to this shift as format collapse.

For Qwen-2.5 7B, answer-only generations are also less accurate than full reasoning-format generations (84.9% vs. 97.6%). This connects the looser syntax reward to the accuracy loss: the policy spends probability mass on the easier accepted format, even when that format removes useful intermediate reasoning. Appendix C.1 gives the full distributional analysis.

Table 4: Format collapse under the composite reward on GSM8K. After training, every evaluated model assigns at least 95.7% of generations to the answer-only shortcut; percentages are means over 3 seeds.

Model	**Ans. only**	**Think+ans.**	**Other**
Gemma-3 4B	97.3 ± 0.4	2.1 ± 0.6	0.6 ± 0.1
Qwen-2.5 7B	98.6 ± 0.3	0.8 ± 0.5	0.6 ± 0.3
Llama-3.1 8B	96.4 ± 0.3	2.7 ± 0.7	0.9 ± 0.5
Ministral 8B	98.1 ± 0.2	1.2 ± 0.4	0.7 ± 0.0
Yi 6B	95.7 ± 0.7	3.2 ± 0.5	1.1 ± 0.3
Gemma-3 12B	98.6 ± 0.2	0.6 ± 0.2	0.8 ± 0.4

Figure 3: Weighted nested rewards. The hard format f_3 is worth 4× the easy format f_1, but early updates still first improve f_1; reward scale changes later learning more than the first format learned.

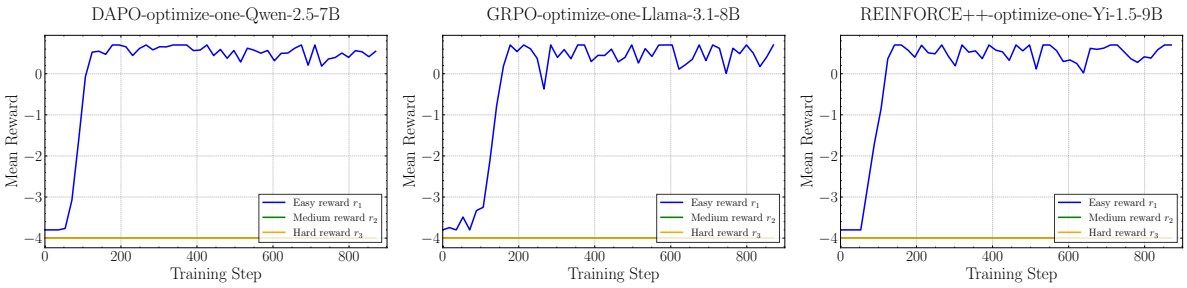

Figure 4: **Mutually exclusive format rewards.** The policy can satisfy exactly one of three disjoint formats, all with equal reward. Across the plotted model-algorithm combinations, it converges to the easiest format, f_1.

5 Characterizing Least-Resistance Learning

Format collapse raises a second question: does PO refuse complex formats, or does it learn them only after easier requirements are already reliable? To answer this, we move from a disjunctive reward to an additive reward over several format requirements. A policy that satisfies only the easiest requirement still loses reward on the harder ones, so the optimal response is to satisfy all requirements.

5.1 Sequential Optimization in Nested Reward Hierarchies

We construct $R_{\text{hierarchy}}$ from three format requirements with increasing structural complexity:

1. **Easy Format (f_1):** Requires the generation to contain a final answer enclosed in `\boxed{.*}`.
2. **Medium Format (f_2):** A superset of f_1, requiring the entire response to be enclosed in `<answer>` tags.
3. **Hard Format (f_3):** A superset of f_2, requiring a full `<think><answer>` structure.

The corresponding languages are nested as $\mathcal{F}_3 \subset \mathcal{F}_2 \subset \mathcal{F}_1$: satisfying f_3 also satisfies f_2 and f_1. This creates a structural incentive to learn the single response pattern that obtains all three rewards. To isolate complexity, we first set all three rewards to the same value, $R_{\max}$.

Figure 2 shows a consistent order. The reward for f_1 rises first, often saturating within 200-300 steps. Progress on f_2 begins after f_1 is reliable. The hardest format, f_3, rises last. The final policy can eventually satisfy the full format (Table 5), but the trajectory is ordered by learnability rather than by the total reward available from the full nested pattern.

This staged behavior supports a least-resistance interpretation: the optimizer first uses the reward channel that becomes learnable earliest, then builds toward harder format components when the easier ones are already stable. The final distribution matters because it separates the path from the endpoint. In the nested setting, the final Qwen-2.5 7B policy mostly reaches f_3, so the claim is not that policy optimization cannot learn the hard format. The claim is that it reaches that format through an easy-to-hard sequence.

5.2 Quantifying Reward Incentives

We next ask whether a larger reward can change the early ordering.

We keep the same nested formats but set the reward magnitudes as a geometric progression: $r(y, f_2) = 2r(y, f_1)$ and $r(y, f_3) = 4r(y, f_1)$.

Figure 3 shows that the larger reward does not

Table 5: Endpoint of the nested hierarchy experiment on Qwen-2.5 7B. The policy eventually reaches the hardest format in most generations, showing that the main claim is about the easy-to-hard learning path rather than an inability to learn the full format; percentages are means over 3 seeds.

Final policy output format	% generations
f_1-only compliant	2.4 ± 0.8
f_2-compliant	6.7 ± 1.2
f_3-compliant	90.9 ± 0.3
Invalid / other	2.1 ± 0.5

Table 6: Endpoint of the mutually exclusive choice experiment on Qwen-2.5 7B. Because each response can match at most one equally rewarded format, the 99.6% share for f_1 isolates the choice of the easiest language; percentages are means over 3 seeds.

Final policy output format	% generations
f_1-compliant	99.6 ± 0.2
f_2-compliant	0.0 ± 0.0
f_3-compliant	0.0 ± 0.0
Invalid / other	0.4 ± 0.1

change the first phase of learning. For roughly the first 150-200 steps, the easiest format improves while the medium and hard formats remain near their initial values. After the easy format is reliable, the medium and hard curves begin to rise together. The larger reward therefore affects later learning, but it does not prevent the optimizer from first taking the easiest available path.

5.3 Mutually Exclusive Rewards: Isolating the Behavior

The nested experiments still share substructure. To test whether the effect remains without shared regex components, we define mutually exclusive formats with identical rewards.

The reward R_{conflict} lets a generation satisfy f_1, f_2, or f_3, but not more than one. The regex languages are pairwise disjoint, and all matched formats receive the same reward.

Let $r(y, f_i)$ be the reward for a generation y satisfying format f_i. The reward function is defined as:

$$r(y, f_i) = R_{\text{max}}, \quad \text{if } y \text{ exclusively matches } f_i$$
$$r(y, f_j) = 0, \quad \text{for all } j \neq i$$

where R_{max} is a constant. The optimizer therefore chooses among three paths with the same payoff but different operational difficulty. Appendix B.1.3 gives the exact disjoint regexes.

Figure 4 shows that every run converges to f_1. The medium and hard reward curves stay near their initial negative values, and the Qwen-2.5 7B final policy produces f_1 in $99.6 \pm 0.2\%$ of generations (Table 6). This rules out the explanation that the nested result is only an artifact of shared substructure. In this experiment, there is no additive incentive to progress from f_1 to f_2 or f_3; all three formats receive the same payoff and cannot be satisfied together. The convergence on f_1 therefore reflects a direct choice among equally rewarded languages.

5.4 Sample-Gated On-Policy Updates

The controlled results are consistent with a simple on-policy gradient account. The expression below abstracts away GRPO/DAPO baselines, clipping, and normalization to isolate the sampling gate. For a format language $\mathcal{F}_i$ with reward weight r_i, the minibatch contribution from m sampled completions is

$$\hat{g}_i = \frac{r_i}{m} \sum_{k=1}^{m} \mathbf{1}\{y_k \in \mathcal{F}_i\} \nabla_\theta \log \pi_\theta(y_k). \quad (3)$$

Let $p_i(\theta) = \Pr_{y \sim \pi_\theta}[y \in \mathcal{F}_i]$ and let N_i be the number of sampled completions in the minibatch that match $\mathcal{F}_i$. Then the number of usable samples and their reward-weighted count satisfy

$$\begin{aligned} \mathbb{E}[N_i] &= m p_i(\theta), \\ \mathbb{E}[r_i N_i] &= m r_i p_i(\theta), \\ \Pr[N_i = 0] &= (1 - p_i(\theta))^m. \end{aligned} \quad (4)$$

Thus, for two formats a and b, format b receives less expected reward-weighted sample mass whenever $r_b p_b(\theta) < r_a p_a(\theta)$. A $4\times$ hard-format reward is outweighed when hard-format support is more than $4\times$ lower, and if $N_b = 0$ there is no positive update for that format. Current policy support gates reward scale.

This is not a convergence theorem, but it accounts for the observed ordering in the controlled settings. In the nested hierarchy, $\mathcal{F}_1$ has the largest support because $\mathcal{F}_3 \subset \mathcal{F}_2 \subset \mathcal{F}_1$, so it supplies positive samples first. In the weighted hierarchy, multiplying the hard-format reward changes payoff but not the chance that a minibatch contains a hard-format sample. In the mutually exclusive setting, all rewards are equal and no response can satisfy two formats, so the earliest reliable positive samples come from the easiest language and on-policy

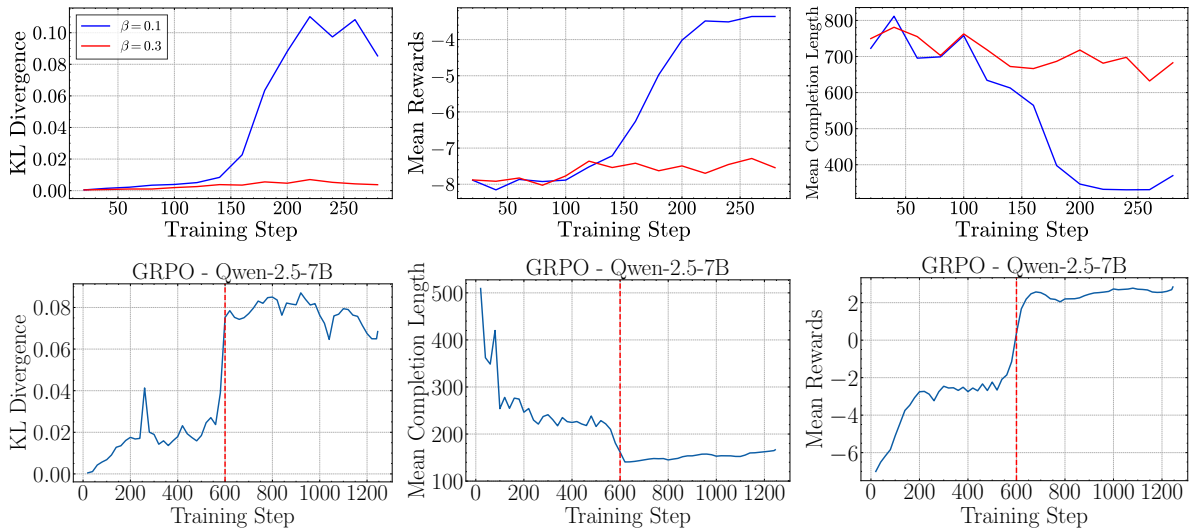

Figure 5: KL diagnostics for format collapse. **Top Row:** A lower KL penalty ($\beta = 0.1$) permits larger policy movement and reaches the answer-only shortcut, while a higher penalty ($\beta = 0.3$) does not. **Bottom Row:** In a single run, KL rises around step 600 as completion length drops and format reward increases.

sampling amplifies that choice. The KL experiments test reachability: a tighter KL penalty limits the policy movement needed for this feedback loop to take over.

6 Exploration Enables the Shortcut

The controlled experiments show what the policy learns. We now ask when the shortcut becomes reachable. Recent RLVR algorithms often reduce or remove KL penalties to let the policy move farther from the reference model (Yu et al., 2025; Yue et al., 2025; Liu et al., 2025c). We use KL as a diagnostic for this movement.

6.1 KL Divergence as a Diagnostic for Exploration

KL divergence measures deviation from the reference policy, π_{ref}. We compare two composite-reward runs that differ only in the KL coefficient β: a lower-penalty run with $\beta = 0.1$ and a higher-penalty run with $\beta = 0.3$.

The lower-penalty policy reaches higher KL and converges to the short answer-only format; the higher-penalty policy stays close to the reference policy and does not discover the same shortcut (Figure 5, top). In a longer GRPO run, the transition is visible around step 600: KL rises, completion length drops, and format reward increases (Figure 5, bottom). Together with Table 4, this suggests that the answer-only shortcut requires a nontrivial policy move from initialization.

This result should not be read as a recommendation to keep KL penalties high. Low KL freedom is often necessary for learning useful reasoning behavior (Liu et al., 2025a). The point is narrower: if a reward admits a cheap syntactic path, exploration can make that path easier to discover and exploit. Reward designers should therefore make the intended reasoning behavior not only high reward, but also the easiest reliable way to obtain that reward.

7 Conclusion

We studied a simple but consequential question in RLVR: when the reward accepts several output formats, which one does policy optimization learn? In the four-benchmark comparison and the GSM8K/controlled-format audits, policies move toward the shortest valid format, often omitting explicit reasoning and reducing task accuracy.

The controlled experiments show why this matters for reward design. The least learnable format is not necessarily the one with the highest reward, and even a 4× reward advantage does not change the early learning order. With disjoint and equally rewarded formats, the policy still converges to the easiest format. A sample-gated gradient account describes this as an on-policy feedback loop: formats receive positive updates only after the current policy samples them, and the expected reward-weighted signal scales with both reward and current-policy support. KL-controlled runs further show that the shortcut appears only when the policy can move far enough from initialization. For RLVR, the intended behavior must be rewarded and easy to reach.

8 Limitations

Our work provides empirical evidence for least-resistance behavior in RLVR format rewards, but its scope is limited in several ways.

Model Scale and Access. Our experiments use public models up to 12B parameters. The trend is consistent across this range, but larger models may find explicit reasoning formats easier to learn, which could change the dynamics.

Algorithmic Diversity. We test GRPO, DAPO, and REINFORCE++, all of which are on-policy policy-gradient methods. Offline RL, DPO-style objectives, supervised fine-tuning, and evolutionary search may show different behavior.

Focus on Empirical Characterization. The paper is empirical. We use learning speed, final format distributions, KL movement, and a sample-gated policy-gradient account as evidence for learnability-driven behavior, but we do not provide a formal convergence analysis.

Computational Constraints on Ablations. Each run required about 8.5 GPU hours (Appendix B.1.2), so we did not run exhaustive sweeps. In particular, the KL analysis compares two coefficients; a denser sweep would better map the exploration trade-off.

9 Ethical Considerations

The main ethical implication is transparency. If a training reward accepts answer-only outputs, optimization may remove explicit reasoning even when the reasoning trace is useful for oversight.

This is a form of reward hacking at the level of output specification: the model satisfies the literal format reward while avoiding the behavior the designer expected. In high-stakes settings, such shortcuts could make outputs harder to audit.

We do not advocate exposing hidden chain-of-thought in deployed systems. The result instead argues for reward and evaluation designs that preserve whatever human-auditable evidence is required for the application, whether that evidence is a private rationale, a verifiable derivation, a tool trace, or another structured artifact.

References

01. AI, :, Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Guoyin Wang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, and 14 others. 2025. Yi: Open foundation models by 01.ai. *Preprint*, arXiv:2403.04652.

Guanzheng Chen, Xin Li, Michael Qizhe Shieh, and Lidong Bing. 2025. Longpo: Long context self-evolution of large language models through short-to-long preference optimization. *Preprint*, arXiv:2502.13922.

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

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3416 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Google DeepMind. 2025. Gemini 3 Pro. https://deepmind.google/models/gemini/pro/. Accessed: 2026-01-04.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. 2025. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models. *Preprint*, arXiv:2501.03262.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego

de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Zehao Jin, J. Lin, and Siao-Fong Li. 2020. Learning principle of least action with reinforcement learning. *ArXiv*, abs/2011.11891.

Shuhei Kurita and Anders Søgaard. 2019. Multi-task semantic dependency parsing with policy gradient for learning easy-first strategies. In *Annual Meeting of the Association for Computational Linguistics*.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. 2025a. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *Preprint*, arXiv:2505.24864.

Yao Liu, Yannis Flet-Berliac, and E. Brunskill. 2022. Offline policy optimization with eligible actions. *ArXiv*, abs/2207.00632.

Yifei Liu, Li Lyna Zhang, Yi Zhu, Bingcheng Dong, Xudong Zhou, Ning Shang, Fan Yang, and Mao Yang. 2025b. rstar-coder: Scaling competitive code reasoning with a large-scale verified dataset. *Preprint*, arXiv:2505.21297.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025c. Understanding r1-zero-like training: A critical perspective. *Preprint*, arXiv:2503.20783.

Harry Mead, Clarissa Costen, Bruno Lacerda, and Nick Hawes. 2025. Return capping: Sample-efficient cvar policy gradient optimisation. *ArXiv*, abs/2504.20887.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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

Sungryull Sohn, Sungtae Lee, Jongwook Choi, H. V. Seijen, Mehdi Fatemi, and Honglak Lee. 2021. Shortest-path constrained reinforcement learning for sparse reward tasks. In *International Conference on Machine Learning*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and yelong shen. 2025. Reinforcement learning for reasoning in large language models with one training example. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. 2025. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *Preprint*, arXiv:2506.14245.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. 2024. On memorization of large language models in logical reasoning.

Roy Xie, David Qiu, Deepak Gopinath, Dong Lin, Yanchao Sun, Chong Wang, Saloni Potdar, and Bhuwan Dhingra. 2025. Interleaved reasoning for large language models via reinforcement learning. *Preprint*, arXiv:2505.19640.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. *Preprint*, arXiv:2002.04326.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, Xiangpeng Wei, Xiangyu Yu, Gaohong Liu, Juncai Liu, Lingjun Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, and 8 others. 2025. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *Preprint*, arXiv:2504.05118.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. 2025. The surprising effectiveness of negative reinforcement in LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

A Appendix

B LLM Usage Statement

The scientific contributions of this paper, including the initial hypothesis, experiment design, and analysis, are the authors' work. The authors used LLMs as assistive tools during manuscript preparation and software development.

Manuscript Preparation. The authors used Google's Gemini 3 Pro Preview (Google DeepMind, 2025) and Gemini 2.5 Pro (Comanici et al., 2025) to help revise wording, improve sentence flow, correct grammar, and fix some LaTeX formatting. All scientific claims remained under author control. The authors reviewed and edited every accepted suggestion.

Code and Plot Generation. LLMs also assisted with boilerplate for data-processing scripts and Python plotting code, including Matplotlib and Seaborn scripts used to visualize the experimental data. The analysis logic and the data were produced by the authors.

Human Oversight. The authors reviewed and verified all text and code that used LLM assistance. The authors are responsible for the final paper.

B.1 Experimental Setup

B.1.1 Optimization Hyperparameters

To reduce training instability, we used a small **learning rate of $5e^{-6}$ for GRPO and DAPO**. For the actor-only **REINFORCE++**, we used $5e^{-5}$. Across optimizers, we used a **cosine annealing scheduler with a 0.1 warmup ratio and a max grad norm of 0.15**. These settings were chosen to make the policy updates stable over the full training horizon.

For **GRPO**, we used the "Done Right" (dr_grpo) variant. Standard GRPO can be sensitive to response length and question difficulty. The dr_grpo loss incorporates corrections for these factors, reducing the chance that the observed format effects are artifacts of these known biases.

Our implementation of **DAPO** builds on this GRPO foundation and uses decoupled clipping. We set $\epsilon_{high} = 0.28$, giving the importance ratio a larger ceiling when increasing the probability of low-probability tokens. This setting is intended to reduce entropy collapse and support stable long-horizon runs.

All training runs used LoRA adapters. We targeted all linear projection layers with rank $\mathbf{r} = 64$ and alpha $\alpha = 128$, which made the runs cheaper than full fine-tuning while leaving enough adapter capacity for the format changes studied here. For REINFORCE++, which requires a separate reference model, the policy and reference used different LoRA adapter heads on the same base model.

B.1.2 Computational Resources

All experiments reported in this paper were run on a private NVIDIA GPU cluster with three RTX A6000 GPUs, each with 48GB of VRAM, and one H100 GPU with 80GB of VRAM. We used `bf16` precision and gradient checkpointing for the larger training runs. A typical run for one model on one dataset required approximately **8.5 GPU hours**. The full suite, including the model, algorithm, and controlled-ablation runs, required approximately **200 GPU hours**. We report these figures to make the compute requirements explicit.

B.1.3 Reward Functions

The experiments combine a *Correctness Reward* with one or more *Format Rewards*. We define these rewards below. Let y be a model generation for a given prompt, and let a be the gold answer.

Correctness Reward (R_{correct}) The correctness reward requires the model's output to be parsable and equivalent to the gold answer. We first define an extraction function, ExtractBoxed(y), which retrieves the content within the final `\boxed{}` environment in y. The correctness reward is:

$$R_{\text{correct}}(y, a) = \begin{cases} +3.0 & \text{Verify_and_parse}(e, a) \\ -5.0 & \text{otherwise} \end{cases} \tag{5}$$

where Verify_and_parse$(\cdot, \cdot)$ checks parsability and mathematical equivalence between the extracted response and the gold answer.

Composite Format Reward ($R_{\text{composite}}$) For the composite-reward experiment, the format reward accepts either an interleaved think/answer structure or a direct `answer`-only format. Let $\mathcal{F}_{\text{composite}}$ be the set of strings matching this format language. The reward is:

$$R_{\text{composite}}(y) = \begin{cases} +3.0 & \text{if } y \in \mathcal{F}_{\text{composite}} \\ -4.0 & \text{otherwise} \end{cases} \tag{6}$$

Composite Regex:

```
^(?:(?:<think>.+?<\/think>
<answer>.+?<\/answer>)+|
(?:<answer>.+?<\/answer>
(?:<think>.+?<\/think><answer>
.+?<\/answer>)*))$
```

The total reward for this experiment is a linear combination of the correctness and composite format rewards.

Strict Format Reward (R_{strict}) For the baseline and control conditions, the format reward accepts only one reasoning structure, following the standard think-then-answer setup used in prior work (DeepSeek-AI et al., 2025). A positive reward is provided only if y matches the `think-then-answer` format below. The reward magnitudes match $R_{\text{composite}}$.

Strict Regex:

```
^<think>.*?</think>
<answer>.*?</answer>$
```

Hierarchical Format Rewards ($R_{\text{hierarchy}}$) For the hierarchy experiments, we use three nested format requirements, f_1, f_2, f_3, with increasing structural complexity.

- **Easy Format (f_1):** The innermost requirement, satisfied if the generation contains a boxed answer. Let $\mathcal{F}_1$ be the set of strings matching `^.*\\boxed\{.+\}.*$`.

$$r(y, f_1) = \begin{cases} +1.0 & \text{if } y \in \mathcal{F}_1 \\ -4.0 & \text{otherwise} \end{cases} \tag{7}$$

Regex:

Easy: `^.*\\boxed\{.+\}.*$`

- **Medium Format (f_2):** A superset of f_1, requiring the boxed answer to also be enclosed within `<answer>` tags.

$$r(y, f_2) = \begin{cases} +2.0 & \text{if } y \in \mathcal{F}_2 \\ -4.0 & \text{otherwise} \end{cases} \tag{8}$$

Regex:

Medium:
`^.*<answer>.*\\boxed\{.+\}.*</answer>.*$`.

- **Hard Format (f_3):** The most encompassing format and a superset of f_2, requiring the full `<think><answer>` structure.

$$r(y, f_3) = \begin{cases} +4.0 & \text{if } y \in \mathcal{F}_3 \\ -4.0 & \text{otherwise} \end{cases} \tag{9}$$

Regex:

Hard: `^<think>.*</think><answer>.*\\boxed\{.+\}.*</answer>$`.

For the "Reward Hierarchy" experiment, the reward magnitudes for $r(y, f_1)$, $r(y, f_2)$, and $r(y, f_3)$ were all set to R_{max}, isolating the effect of structural complexity. For the weighted hierarchy experiment, the scaled values were 1.0, 2.0, and 4.0. The total reward for these experiments, $R_{\text{hierarchy}}(y)$, is the sum of the rewards for all satisfied format conditions, so a generation matching f_3 also receives the rewards for f_2 and f_1. For the "Mutually Exclusive Choice" experiment, the total reward is the reward for the single format that is exclusively matched.

Reward Aggregation for Hierarchical vs. Mutually Exclusive Experiments

The distinction between the *Reward Hierarchy* experiments and the *Mutually Exclusive Choice* experiment is how the final scalar reward is calculated from the individual format conditions.

In the *Composite Reward experiment* (Section 4.2), the reward function $R_{\text{composite}}(y)$ is **disjunctive**: it gives one constant positive reward if y matches *any* valid format in Regex B.1.3. To obtain the maximum format reward, the optimizer can satisfy any one of the accepted regex conditions. The observed convergence to the `answer`-only format is the format collapse studied in the main paper.

In the hierarchy experiments, the total reward, $R_{\text{hierarchy}}(y)$, is the **sum of the rewards for all satisfied format conditions**. A generation y that satisfies the most complex format, f_3, also satisfies the nested sub-formats f_2 and f_1. Its total reward is:

$$R_{\text{total}}(y) = r(y, f_1) + r(y, f_2) + r(y, f_3) \quad \text{if } y \in \mathcal{F}_3. \tag{10}$$

This additive structure gives the optimizer a persistent incentive to progress through the hierarchy. A policy that converges only to f_1 or f_2 continues to incur penalties from the unsatisfied, more complex format conditions. The setup therefore lets

us observe the *path* the optimizer takes when the highest-reward endpoint is the full f_3 format.

In the *Mutually Exclusive Choice* experiment, the total reward, $R_{\text{conflict}}(y)$, is the reward for the **single format that is exclusively matched**. Start- and end-of-string anchors in the regular expressions ensure that a generation can satisfy at most one format condition. The reward function is:

$$R_{\text{total}}(y) = \max\left(r(y, f_1), r(y, f_2), r(y, f_3)\right)$$

where only one term can be non-zero. This setup removes the additive incentive to progress and eliminates shared regex substructure. Its convergence on f_1 shows that the nested result is not only an artifact of incremental learning over shared components.

Conflicting Format Rewards (R_{conflict}) For the "Mutually Exclusive Choice" experiment, the format requirements are pairwise disjoint, so a single generation y cannot satisfy more than one condition. This removes shared regex sub-problems and creates a direct choice between formats. The reward function, $R_{\text{conflict}}(y)$, provides a constant positive reward R_{max} if and only if y exclusively matches one of the following formats: f_1, f_2, or f_3.

- **Easy Format (f_1):** Defined by the prior Easy Regex B.1.3. This format is satisfied if the generation ends *immediately* after the closing brace of a boxed{} environment. The `$` anchor asserts that no characters, including the `</answer>` tag, can follow the boxed answer.

- **Medium Format (f_2):** Defined by the regex:

  Regex:

  Medium: `^(?:<answer>.+?\\boxed\{.+\}<\/answer>)$`

  This format requires the entire generation to be a single, valid `<answer>` block that contains a boxed answer. The ˆ and **$** anchors require the generation to begin with `<answer>` and end with `</answer>`, with no preceding `<think>` tags.

- **Hard Format (f_3):** Defined by the regex:

  Regex:

  Hard: `^<think>.+<\/think><answer>.+\\boxed\{.+\}<\/answer>$`

  This format requires the strict *think-then-answer* structure. The ânchor requires the generation to start with a `<think>` tag.

Justification for Mutual Exclusivity: The formats are disjoint because of the start-of-string ˆ and end-of-string '$' anchors in the regular expressions:

❶ A generation matching f_3 cannot match f_2 because it must start with `<think>`, whereas f_2 must start with `<answer>`.

❷ A generation matching f_2 cannot match f_1 because it must end with `</answer>`, whereas f_1 must end immediately after the } of the boxed expression.

❸ Similarly, a generation matching f_3 cannot match f_1 because it must end with `</answer>` rather than immediately after the boxed expression.

These definitions make the three reward paths independent at the regex level. The resulting convergence on f_1, as shown in the main paper, cannot be explained by shared format substructure.

B.2 System Prompts

The system prompt frames the task for the model and can influence its behavior. We therefore used prompts that define the valid formats without explicitly ranking them. Below, we list the prompts used for the core experiments and explain the relevant control.

Prompt for Composite and Hierarchical Reward Experiments For all experiments involving a choice between multiple valid formats (Section 4.2), we used the following system prompt. It defines the available tags (`<think>`, `<answer>`, `\boxed{}`) without saying that one valid format is preferable to another.

System Prompt:

You are a helpful assistant, and you will be providing factually correct responses to the user queries. While responding, if at any point you feel like you are not comfortable answering the question, you can choose to think in an internal dialogue and enclose your thoughts between `<think></think>` tags, and then when you are confident again, you can resume answering by enclosing your answer between `<answer></answer>` tags. You can choose to think whenever you feel appropriate, and then answer when you are confident. Note that this also means that you can think a bit, answer that portion, and start thinking again when you need to, which means you can have interleaving `<think>...</think>` and `<answer>...</answer>` tags in your response. Enclose your final response to the query in `\boxed{}`.

Design Rationale: The key property of this prompt is that it is permissive but not ranked. Phrases such as *you can choose to think*, *whenever you feel appropriate*, and the explicit mention of an "interleaving" structure make the composite and hierarchy formats available. The prompt provides no explicit instruction to prefer a minimal response.

Prompt for Strict think-then-answer Baseline For our baseline experiments and the "Mutually Exclusive Choice" experiment (where f_3 was the hard format), we used a standard, more restrictive prompt.

System Prompt:

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first needs to think between `<think>...</think>`, and then respond between `<answer>...</answer>` tags. The final response within the answer must be enclosed within `\boxed{}`.

Design Rationale: This prompt is prescriptive. The phrase *first needs to think... and then respond* requires a single sequential format. This serves as the strict-format control against which we compare policies trained with the composite reward. The comparison keeps the intended format explicit in the strict condition and non-ranked in the composite condition.

C Policy Analysis

C.1 Quantitative Evidence for Format Collapse

Section 4.2 describes format collapse: policies trained with the composite reward function ($R_{\text{composite}}$) converge to the simplest valid output format. This section provides the supporting distributional data.

To measure the final policy's learned format distribution, we generated 600 samples from the GSM8K test set for each trained model. Each generation was categorized with the same regex-based classifier used in the main analysis. The classification ignores answer correctness so that it isolates output syntax.

Table 4 presents the results. For every model, the `<answer>`-only format accounts for more than 95% of generated outputs, with Qwen-2.5 7B and Gemma-3 12B reaching 98.6%. The full reasoning format appears in less than 4% of generations for every model.

The key takeaway is that format collapse is not an occasional artifact in these runs. Under a single reward function that accepts multiple formats, the optimizer places almost all mass on the shortest accepted format.

C.1.1 Policy Distributions for Controlled Experiments

The learning curves in Section 5 (Figures 2 and 4) show mean reward over time. This section adds final output-format distributions for the same controlled settings. All data are from Qwen-2.5 7B, averaged over 3 random seeds.

Reward Hierarchy Experiment. Table 5 gives the final policy distribution from the nested reward experiment. The final policy is mostly compliant with the most complex format, f_3, which accounts for approximately 91% of generated outputs.

This outcome does not contradict the least-resistance interpretation. As explained in Section 5, the total reward in this experiment is the sum of the rewards for each satisfied format, so the optimizer has a persistent incentive to reach the full format. The key evidence is the path to that final state, shown in Figure 2.

The optimizer does not learn the unified f_3 objective directly from the start. It first learns the simplest component (f_1), then the intermediate component (f_2), and only later converges on the final format (f_3). This behavior differs from the convergence on f_1 in the mutually exclusive experiment (Table 6), where the incentive to progress is absent.

Mutually Exclusive Choice Experiment. Table 6 presents the final policy distribution for the experiment with disjoint, equally rewarded formats. More than 99% of generations from the final policy match the simplest format, f_1. This shows that the nested-result interpretation cannot rely only on shared substructure.

Table 7: **Full Performance Comparison with Confidence Intervals.** We report final accuracy (%) for models trained with a **Strict Reward** versus a **Composite Reward**. This is the detailed version of Table 1. Confidence intervals ($\pm$) are standard errors over 3 random seeds. Strict formatting wins most model-task pairs, while the composite reward sometimes matches or exceeds it on Gemma-3 4B GSM8K, Qwen-2.5 7B rStar-Coder, and Ministral 8B Planning-Mystery. Best performance in each pair is highlighted in bold.

	GSM8K		**rStar-Coder**		**ReClor**		**Planning-Mystery**	
Model	**Strict**	**Composite**	**Strict**	**Composite**	**Strict**	**Composite**	**Strict**	**Composite**
Gemma-3 4B	72.4 ± 0.5	**73.1** ± 0.4	**55.8** ± 0.6	53.5 ± 0.7	**73.2** ± 0.3	69.4 ± 0.5	**58.3** ± 0.4	55.9 ± 0.6
Qwen-2.5 7B	**92.4** ± 0.3	85.5 ± 0.5	73.2 ± 0.4	**73.6** ± 1.2	**80.1** ± 0.2	73.8 ± 0.4	**77.5** ± 0.3	73.8 ± 0.5
Llama-3.1 8B	**86.2** ± 0.4	82.9 ± 0.6	**64.1** ± 0.5	59.2 ± 0.6	**78.3** ± 0.3	72.7 ± 0.5	**74.9** ± 0.4	70.6 ± 0.7
Ministral 8B	**89.5** ± 0.3	84.8 ± 0.5	**74.0** ± 0.4	72.6 ± 0.4	**81.0** ± 0.2	76.5 ± 0.4	76.8 ± 0.3	**77.3** ± 0.3
Yi 6B	**85.1** ± 0.4	83.8 ± 0.4	**68.3** ± 0.6	63.7 ± 0.7	**68.4** ± 0.4	65.1 ± 0.6	**71.2** ± 0.5	67.7 ± 0.6
Gemma-3 12B	**94.6** ± 0.2	86.4 ± 0.4	**76.1** ± 0.4	72.3 ± 0.5	**85.5** ± 0.2	81.2 ± 0.3	**81.2** ± 0.3	76.5 ± 0.5

User

Half the value of $3x - 9$ is $x + 37$. What is the value of x?
Answer: 83

Qwen-7B $\beta \to 0.1$

$r_{\text{composite}}$

<answer>The equation is $\frac{3x-9}{2} = x + 37$. Multiplying by 2 gives $3x - 9 = 2x + 74$. Subtracting $2x$ from both sides gives $x - 9 = 74$. Adding 9 to both sides gives $x = \boxed{83}$.</answer>

Figure 6: Low-KL composite example on a GSM8K algebra problem. The generation uses the direct `<answer>`-only format and reaches the gold answer.

User

You have seven bags of gold coins. Each bag has the same number of gold coins. One day, you find a bag of 53 coins. You decide to redistribute the number of coins you have so that all eight bags you hold have the same number of coins. You successfully manage to redistribute all the coins, and you also note that you have more than 200 coins. What is the smallest number of coins you could have had before finding the bag of 53 coins?
Answer: 203

Qwen-7B $\beta \to 0.1$

$r_{\text{composite}}$

<answer>Let N be the coins in each of the seven bags. The total is $7N + 53$. This must be a multiple of 8 and greater than 200. Testing values for N where $N \equiv 5 \pmod 8$, the first value where $7N + 53 > 200$ is $N = 27$. The original number of coins was $7 \times 27 = \boxed{189}$.</answer>

Figure 7: Low-KL composite example on a GSM8K divisibility problem. The response follows the accepted answer-only format but gives the wrong final answer, illustrating that format validity and answer correctness are evaluated separately.

User

Half the value of $3x - 9$ is $x + 37$. What is the value of x?
Answer: 83

Qwen-7B $\beta \to 0.3$

$r_{\text{composite}}$

<think> The equation is set up as... </think> $3x - 9 = 2(x + 37)$ <answer> solving for x gives </answer> x is 83.

Figure 8: Higher-KL-penalty composite example on the same algebra prompt as Figure 6. The output stays closer to a mixed reasoning format instead of collapsing to a clean answer-only response.

User

You have seven bags of gold coins. Each bag has the same number of gold coins. One day, you find a bag of 53 coins. You decide to redistribute the number of coins you have so that all eight bags you hold have the same number of coins. You successfully manage to redistribute all the coins, and you also note that you have more than 200 coins. What is the smallest number of coins you could have had before finding the bag of 53 coins?
Answer: 203

Qwen-7B $\beta \to 0.3$

$r_{\text{composite}}$

Let N be the original number of coins. <answer> 7N + 53 </answer> <think> must be divisible by 8. And > 200. </think> The answer is 711.

Figure 9: Higher-KL-penalty composite example on the same divisibility prompt as the preceding low-KL example. The output does not take the clean answer-only shortcut and remains syntactically messy.

User

A ship traveling along a river has covered $24\ \mathrm{km}$ upstream and $28\ \mathrm{km}$ downstream. For this journey, it took half an hour less than for traveling $30\ \mathrm{km}$ upstream and $21\ \mathrm{km}$ downstream, or half an hour more than for traveling $15\ \mathrm{km}$ upstream and $42\ \mathrm{km}$ downstream, assuming that both the ship and the river move uniformly.
Determine the speed of the ship in still water and the speed of the river.

Qwen-7B + OpenR1-Math220K

r_{strict}

<think>Let's think step by step...</think> <answer>... $\boxed{4, 10}$</answer>

$r_{\text{composite}}$

<answer>Let's denote the speed of the ship in still water as s and the speed of the river as ... $10 km/h and the speed of the river is 2 km/h. $\boxed{10, 2}$</answer>

Figure 10: Example comparison after OpenR1-Math220K training. The strict reward preserves a think-then-answer response, while the composite reward permits a shorter answer-only response with a different boxed value.

User

Let AB be a chord of the unit circle $\odot O$. If the area of $\odot O$ is exactly equal to the area of the square with side AB, then $\angle AOB =$ (to 0.001 degree).

Qwen-7B + OpenR1-Math220K

r_{strict}

<think>Okay, so the problem is about a chord AB on the unit circle ... So, setting these equal ... </think> <answer>124.806°</answer>

$r_{\text{composite}}$

<answer>First, we need to find the area of the unit circle ... Therefore, the angle AOB is approximately 65.427 degrees, rounded to 0.001 degree. $\boxed{65.427}$</answer>

Figure 11: Geometry example after OpenR1-Math220K training. The strict and composite policies produce different response formats and different final numerical answers.

User

Four boys bought a boat for $60. The first boy paid one half of the sum of the amounts paid by the other boys; the second boy paid one third of the sum of the amounts paid by the other boys; and the third boy paid one fourth of the sum of the amounts paid by the other boys. How much did the fourth boy pay?
Answer: 13

Qwen-7B + OpenR1-Math220K

r_{strict}

<think>Let x_1, x_2, x_3, x_4 be the amounts paid by the four boys. We are given that $x_1 + x_2 + x_3 + x_4 = 60$. The first boy paid one half of the sum of the amounts paid by the other boys, so $x_1 = \frac{1}{2}(x_2 + x_3 + x_4)$. The second boy paid one third of the sum of the amounts paid by the other boys, so $x_2 = \frac{1}{3}(x_1 + x_3 + x_4)$... </think> <answer>The amounts paid by the four boys are $20, 15, 12, 13$. Let's check the conditions: $x_1 = 20$, $x_2 = 15$, $x_3 = 12$, $x_4 = 13$. $x_1 = \frac{1}{2}(x_2+x_3+x_4) = \frac{1}{2}(15+12+13) = \frac{1}{2}(40) = 20$. $x_2 = \frac{1}{3}(x_1+x_3+x_4) = \frac{1}{3}(20+12+13) = \frac{1}{3}(45) = 15$. $x_3 = \frac{1}{4}(x_1 + x_2 + x_4) = \frac{1}{4}(20 + 15 + 13) = \frac{1}{4}(48) = 12$. The conditions are satisfied.
Final Answer: The final answer is $\boxed{13}$</answer>

$r_{\text{composite}}$

<answer>$\boxed{10}$</answer>

Figure 12: Arithmetic example after OpenR1-Math220K training. The strict policy gives a worked think-then-answer response, while the composite policy emits only a boxed answer.