What Makes a Rubric Verifier Usable as an RL Reward? A Criterion-Vector Reward Interface for STEM Reasoning

Anonymous ACL submission

Abstract

Rubrics turn long-form STEM reasoning into dense rewards, but once they enter an RL loop, reward quality hinges on the verifier that maps candidate solutions to criterion-level verdicts. We frame rubric-conditioned verification as a reward-interface problem and identify three requirements for RL use: (1) *Criterion-level faithfulness* requires that the scalar reward must be backed by the correct criterion verdicts; (2) *Surface-form invariance* requires that rephrasing, formatting, rubric-copying, or verbosity should not change rewards when rubric satisfaction is unchanged; (3) *Servable output compliance* requires the verifier to emit complete, concise, parser-valid criterion vectors for rollout-time reward serving. We instantiate this problem with **RUBRIC-HARD-STEM**, a 300-example diagnostic benchmark spanning 2,949 human-reviewed criterion labels, and develop **RUBRICVERIFIER-STEM**, a compact verifier trained through diverse-source SFT, agreement-first RSFT, and interface-aware RL. Initialized from Qwen3-4B-Instruct, the verifier raises macro RUBRICACC from 66.6% to 82.9% and reduces false credit from 35.4% to 10.3%. When served as the reward model, it improves the average score of Qwen3-4B-Thinking on 17 challenging STEM benchmarks from 38.3% to 47.5%.

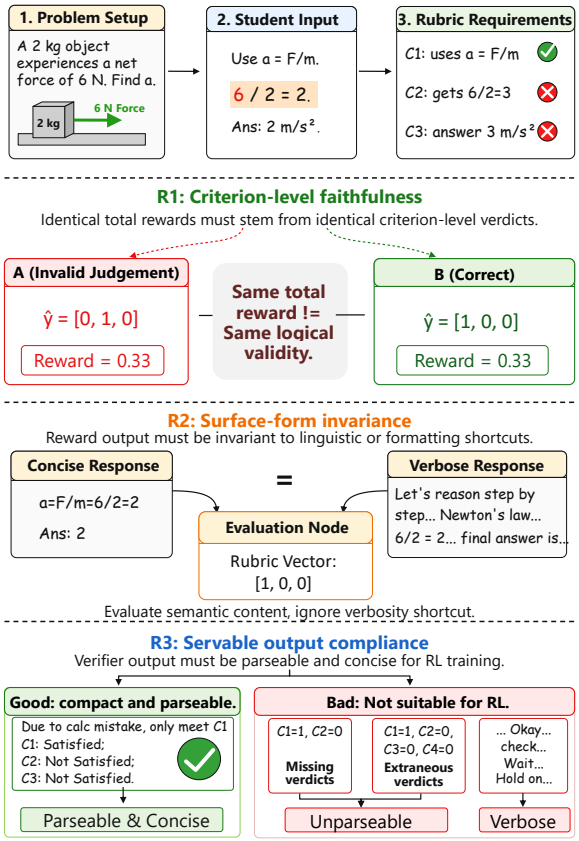

Figure 1: Illustration of STEM rubric grading and three requirements for rubric verifier

1 Introduction

A central challenge in reinforcement learning (RL) for STEM reasoning is designing reward signals that provide informative credit assignment over multi-step solutions, rather than only sparse final-answer feedback (Lightman et al., 2024; Wang et al., 2024; Lu et al., 2024; Luo et al., 2025). Existing work has shown that outcome rewards can improve final-answer accuracy through RL (Cobbe et al., 2021; Liu et al., 2025; DeepSeek-AI et al., 2025), yet they provide only sparse supervision without distinguishing the quality of intermediate reasoning. Rubrics offer a natural source of denser supervision by decomposing solutions into explicit grading criteria, allowing reward models to assign credit to local reasoning components rather than to the final answer (Gunjal et al., 2025; Huang et al., 2025). However, rubric-based RL requires more than appending rubric text to a judge; it depends on a rubric-conditioned verifier that converts rubric criteria into verdicts for subsequent reward scalarization. This raises an underexplored question: *What properties must such a rubric verifier satisfy for RL optimization?*

To address this question, we frame rubric-

conditioned verification as a criterion-vector reward interface and identify three RL-facing requirements. ***(R1) Criterion-level faithfulness*** requires the verifier to assign the right verdict to each rubric clause, not merely to match the final scalar grade. This matters in hard partial-credit cases where valid intermediate reasoning, local errors, and fluent but invalid derivations must be separated. ***(R2) Surface-form invariance*** requires the induced reward to remain stable under reward-irrelevant changes to the candidate answer, such as rephrasing, formatting, rubric-copying, or verbosity padding, when the underlying rubric satisfaction is unchanged. ***(R3) Servable output compliance*** requires the verifier to emit complete, concise, parser-valid criterion vectors that can be repeatedly scalarized during rollout-time reward serving.

To measure this interface, we introduce **RUBRIC-HARD-STEM**, a 300-example diagnostic benchmark for long-form STEM and applied-engineering grading. It contains 2,949 human-reviewed criterion labels across five domains and controlled score regimes. RUBRIC-HARD-STEM turns the requirements into observable diagnostics: false credit and missed credit expose violations of criterion-level faithfulness (R1), paired concise/verbose rewrites expose violations of surface-form invariance (R2), and missing, duplicated, extra, malformed, or overly verbose verifier outputs expose failures of servable output compliance (R3). We then build **RUBRICVERIFIER-STEM**, a compact verifier trained through diverse-source supervised fine-tuning (SFT), agreement-first rejection-sampled fine-tuning (RSFT), and interface-aware RL. Serving RUBRICVERIFIER-STEM as the reward model, policy model initialized from Qwen3-4B-Thinking (Qwen Team et al., 2025) achieves a 47.5% average score across 17 challenging STEM benchmarks through RL, a +9.3-point gain over the no-RL baseline and +3.8 points above the strongest scalar-reward baseline. Our contributions are as follows:

- We formulate rubric-conditioned verification as a criterion-vector reward interface for RL and identify three requirements: criterion-level faithfulness, surface-form invariance, and servable output compliance. We introduce RUBRIC-HARD-STEM to measure violations of these requirements across five STEM domains.
- We propose RUBRICVERIFIER-STEM, a compact verifier that decouples faithful criterion-level judgment from concise reward serving. It improves macro RUBRICACC from 66.6% to 82.9% over Qwen3-4B-Instruct, while reducing false credit, verbosity sensitivity, and rollout cost; the gains also transfer to ProfBench.
- When used as a criterion-vector reward source, RUBRICVERIFIER-STEM improves a Qwen3-4B-Thinking RL policy from 38.3% to 47.5% average STEM score. It outperforms outcome and scalar-reward baselines, and matches a much larger GPT-OSS-120B (OpenAI et al., 2025) criterion-vector verifier within 0.4 points.

2 From Reward Requirements to a Diagnostic Benchmark

This section turns the reward-interface contract into measurable diagnostics. The criterion-vector task defines what must be faithful (R1). Paired surface rewrites test whether the reward changes for non-semantic reasons (R2). Parser-facing diagnostics test whether verifier outputs can be served as rollout-time rewards (R3). RUBRIC-HARD-STEM records the criterion-level decisions behind every scalar grade, making incorrect criterion-level credit, verbosity-sensitive reward shifts, and malformed criterion vectors directly observable. Table 1 maps these diagnostics to R1–R3.

2.1 Criterion-Vector Verification Task

Each benchmark instance consists of a problem x, a candidate solution s, and an instance-specific rubric $R = \{(c_i, w_i)\}_{i=1}^{m}$ with optional nonnegative weights w_i. A verifier predicts a binary vector $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_m)$, where $\hat{y}_i = 1$ means that s satisfies criterion c_i. When a rubric does not provide weights, we set $w_i = 1$.

The scalar grade used by RL is obtained only after these local decisions:

$$\mathrm{Grade}(\hat{\mathbf{y}}, R) = \frac{\sum_{i=1}^{m} w_i \hat{y}_i}{\sum_{i=1}^{m} w_i}. \quad (1)$$

This motivates the criterion-level faithfulness requirement: matching the scalar grade is not sufficient, because two different verdict vectors can induce the same grade while assigning credit to different rubric clauses.

2.2 Benchmark Design Goals

RUBRIC-HARD-STEM is designed to make reward-relevant verifier errors observable. First,

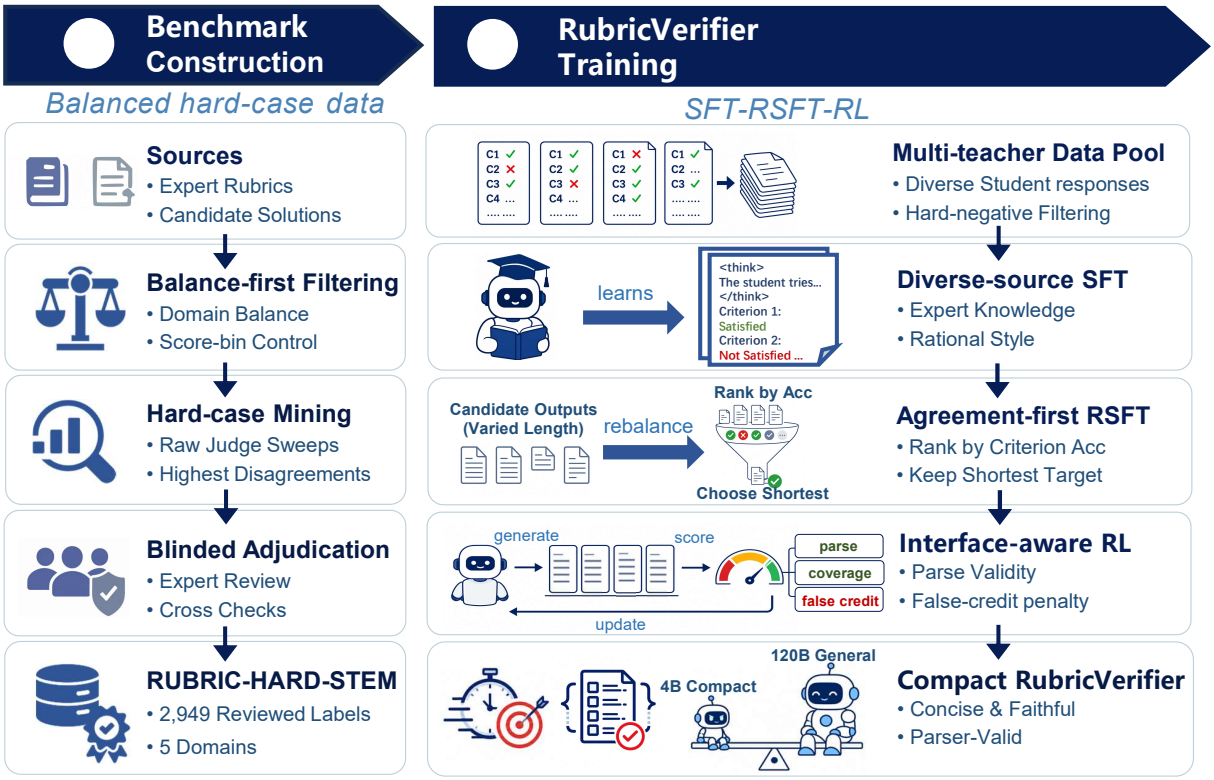

Figure 2: Data and training pipeline for the rubric reward interface. The pipeline separates reliable criterion-label construction from concise rollout-time reward serving.

it covers hard partial-credit regimes for R1: cases where valid intermediate reasoning should receive credit despite a wrong final answer, and cases where fluent but invalid reasoning should be denied criterion-level credit. Second, it tests R2 through paired concise/verbose rewrites whose gold criterion vectors are held fixed, so any reward shift reflects surface-form sensitivity rather than changed rubric satisfaction. Third, it tests R3 by counting missing, duplicated, extra, or unparsable verifier verdicts as servable-output failures.

2.3 Data Sources and Candidate Pool Construction

The benchmark candidate question–rubric pairs are sampled from public resources that already attach rubric, marking-scheme, or criterion-style grading information written by human experts to challenging STEM and professional questions, including HiPhO (Yu et al., 2025), SUPERChem (Zhao et al., 2025), FrontierScience (Wang et al., 2026), IMO-Proof Bench (Luong et al., 2025), and RubricBench (Zhang et al., 2026). Candidate solutions and criterion-verification targets are newly produced in our pipeline. This design combines expert-written rubric quality with model-generated answers that better reflect rubric-based RL use cases.

We build answer pools by pairing these sourced question–rubric pairs with fresh answers from multiple student models. Fixed open-weight teacher verifiers, including GPT-OSS-120B and Gemma-4-31B-it (Google DeepMind, 2026), generate criterion-level judgment candidates and disagreement signals for triage. The answer pool includes fully correct answers, partially correct derivations, plausible but locally invalid answers, and length-controlled rewrites used to test whether verifiers reward superficial elaboration.

2.4 Balance Before Hard-Case Mining

To evaluate criterion-level credit assignment, we first control the balance between satisfied and unsatisfied criteria. Without this control, verifier accuracy can reflect the base rate of rubric satisfaction rather than genuine judgment quality, lenient verifiers benefit when most criteria are satisfied, while strict verifiers benefit when most are unsatisfied. High-disagreement cases are then selected from this balanced pool, ensuring that the diagnostic set probes criterion-level credit-assignment errors rather than satisfaction-frequency artifacts.

After applying these controls, RUBRIC-HARD-

STEM contains 300 problem–response–rubric examples, with 60 examples per domain and 50 examples in each of six satisfied-ratio regimes. Within each domain–score stratum, we prioritize high-disagreement candidates for expert adjudication. Appendix Table 7 gives the domain counts, and Figure 5 visualizes the resulting balance.

2.5 Blinded Expert Annotation and Leakage Control

RUBRIC-HARD-STEM is built with a two-stage process that combines model-assisted case discovery with blinded expert labeling. We first use automated verifier scores and deliberative judge passes to identify candidate answers on which LLM-based verifiers disagree or are likely to be uncertain, since these cases are most informative for evaluating criterion-level verification. The final criterion labels are assigned by three experts who independently inspect the problem, reference solution, rubric, candidate answer, and target criterion, without access to model provenance or automated scores. Labels show high reliability, with 94.0% unanimous agreement and Fleiss' κ=0.92; remaining disagreements are resolved by a senior expert under the same blinding protocol. To prevent leakage, we deduplicate RUBRIC-HARD-STEM against all data used for SFT, RSFT, verifier GSPO, checkpoint selection, prompt tuning, and downstream reward-source selection, ensuring that no RUBRIC-HARD-STEM item appears in any of these stages. Appendix E shows representative RUBRIC-HARD-STEM examples with their criterion-level labels.

2.6 Evaluation Metrics

We primarily evaluate the performance through RUBRICACC, the per-example weighted criterion accuracy macro-averaged across examples, which avoids overweighting longer rubrics. We also report micro criterion accuracy (CRITACC). Scalar agreement is measured by GradeMAE, the mean absolute error between the predicted grade in Equation 1 and the gold grade. Since scalar agreement can mask clause-level errors, we further report SATACC and UNSATACC on gold-satisfied and gold-unsatisfied criteria, together with false-credit and missed-credit rates. Missing, duplicated, extra, or unparsable verdicts are counted as incorrect and reported as servable-output failures. Partially satisfied verdicts are seen as not satisfied.

3 Building a Rollout-Time Rubric Reward Verifier

We design the training pipeline around three diagnostic failures of the Qwen3-4B-Instruct starting point on RUBRIC-HARD-STEM. The base verifier reaches only 66.6% macro RUBRICACC with a 35.4% false-credit rate, indicating limited criterion-level faithfulness (R1); verbose rewrites with unchanged gold labels shift its predicted grade by +5.6 points, revealing surface-form sensitivity (R2); and at rollout time it produces longer outputs and more malformed responses than GPT-OSS-120B, falling short of servable output compliance (R3). RUBRICVERIFIER-STEM addresses these failures in stages: diverse-source SFT broadens criterion-level judgment coverage for R1; agreement-first RSFT distills faithful judgments into concise, parser-valid targets, improving R3 while preserving verdict agreement; and interface-aware RL penalizes surface-induced false credit and parser-facing failures under sampled rollout behavior, targeting R2 and residual R3 errors. Figure 2 and Appendix Table 8 summarize the stage-by-stage safeguards.

3.1 Diverse-Source SFT

The first stage teaches the verifier the task format and exposes it to varied sources of partial correctness while excluding RUBRIC-HARD-STEM items. The SFT rubric/question seeds come from RaR-Science (Gunjal et al., 2025) and RubricHub (Li et al., 2026); the benchmark candidate question–rubric pairs in Section 2 are held out. We construct more than 100K generated verifier examples from multiple student-answer generators, open-weight teacher responses, and filtered hard negatives. These pseudo-target verifier outputs support distillation and hard-case exposure while remaining separate from benchmark gold labels.

Diversity matters because verifier correctness can otherwise become confounded with one generator's answer style, verbosity, or preferred explanation template. At each stage, we rebalance examples by subject, gold satisfied-ratio bins, student-generator source, and the verifiers used for judging, so that the benchmark covers diverse domains, score levels, answer-generation models, and verifier behaviors.

3.2 Agreement-First RSFT

SFT teaches the verifier the grading format and task knowledge, but does not fully close the accu-

Table 1: The reward-interface contract for criterion-vector verification. Each row links a verifier requirement to the RL failure signal and the corresponding static and in-loop tests.

Requirement	RL need	Failure signal	Static test	In-loop test
R1. Criterion-level faithfulness	Correct per-criterion credit.	False or missed credit.	Section 4.1; Table 2; Figure 3.	Section 4.3; blinded rollout audit.
R2. Surface-form invariance	Reward independent of verbosity, formatting, or rubric copying.	Verbose rewrites receive higher grade.	Section 4.2; paired concise/verbose rewrites; Figure 4; Table 3.	Section 4.3; within-prompt length comparison.
R3. Servable output compliance	Complete parseable vectors at rollout rate.	Unparseable verdicts; verbose outputs.	Output length in Table 2; format/serving diagnostics in Table 11.	Section 4.3; throughput and parse-fail rate.

racy gap. We therefore apply *rejection-sampled fine-tuning* (RSFT) to improve criterion-level accuracy while preserving rollout-time compactness. Each sampled output is parsed into a criterion vector and ranked by agreement with the target vector, yielding a 30K+ agreement-filtered RSFT pool. Among parser-valid outputs with the best or near-best agreement, we select the shortest target. Length is therefore used only as a tie-breaker after criterion-level correctness has been established, targeting servable output compliance (R3) without weakening criterion-level faithfulness (R1).

3.3 Hardening the Interface with RL

SFT and agreement-first RSFT establish faithful and compact criterion-vector outputs, but rollout-time serving exposes two remaining risks. The verifier may change its verdicts when only the candidate answer's surface form changes (R2), or it may emit incomplete, malformed, or excessively long outputs that cannot be reliably scalarized (R3). We introduce an RL refinement stage that directly optimizes the sampled verifier behavior used during policy rollouts. We instantiate this stage with RL and focus training on difficult examples where sampled verifier judgments are unstable or repeatedly disagree with the target criterion vector.

The verifier reward combines three terms aligned with the contract: parsed criterion-level accuracy for R1, targeted penalties for length-perturbed false credit for R2, and parser-facing coverage and format penalties for R3. Malformed, missing, duplicated, or extraneous verdicts are penalized, while parse validity and criterion coverage are preserved as requirements for rollout-time reward serving. Appendix Section C.2 gives the full reward decomposition and difficulty-aware rollout gate, Appendix Table 13 gives the scalar coefficients, and Appendix Section C.1 gives the GSPO objective.

4 Experiments

Our experiments follow the three-part reward-interface contract. Section 4.1 evaluates criterion-level faithfulness (R1) on RUBRIC-HARD-STEM and reports whether the resulting verifier remains compact enough for serving (R3). Section 4.2 tests surface-form invariance (R2) with paired concise/verbose rewrites. Section 4.3 then checks whether the same verifier can serve as a live RL reward under matched downstream training. The models are evaluated with 8 responses with temperature 0.7 and reported as the mean.

4.1 Does the Verifier Improve Faithfulness Without Losing Compactness?

This section evaluates criterion-level faithfulness (R1) on RUBRIC-HARD-STEM: whether the verifier assigns the correct verdict to each rubric clause rather than merely matching the aggregate grade. Table 2 shows that RUBRICVERIFIER-STEM substantially improves this operating point. Relative to the Qwen3-4B-Instruct base, it raises macro RUBRICACC from 66.6 to 82.9 and CRITACC from 68.5 to 85.6, while reducing GradeMAE from 27.5 to 14.9. The gain is not simply due to stricter grading: false credit drops from 35.4 to 10.3, while SATACC also improves from 73.5 to 80.6 and UNSATACC reaches 89.7. Figure 3 shows the same pattern across domains, with gains over the base model in all five domains and performance within 3.4 points of GPT-OSS-120B in every domain, slightly surpassing it in math.

The same result also supports the compactness side of servable output compliance (R3). RUBRICVERIFIER-STEM obtains a +16.3 point RUBRICACC gain over the base verifier while increasing visible output length only from 314.8 to 458.8 tokens, rather than relying on long judge-style rationales. It is only 1.0 point below

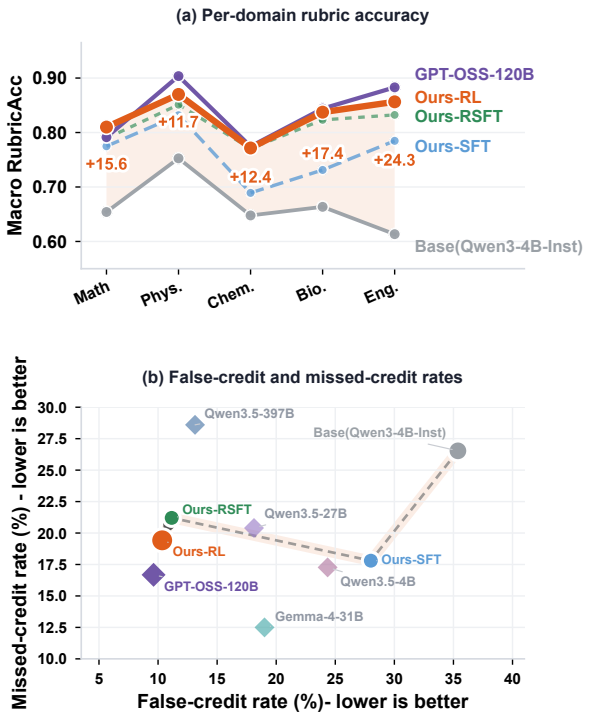

Figure 3: **(A)** Macro RubricAcc by domain for the base model, SFT, RSFT, the final GSPO checkpoint, and GPT-OSS-120B; the shaded band highlights lift from base to GSPO, with per-domain point gains. **(B)** False credit versus false negatives (lower left is better) and the SFT→RSFT→GSPO trajectory.

GPT-OSS-120B in macro RubricAcc while using fewer output tokens, and it outperforms the Qwen3.5 open-weight reference models in both accuracy and compactness. Thus, on RubricHard-STEM, the verifier improves criterion-level faithfulness without sacrificing the concise serving profile needed for rollout-time reward use.

The RSFT ablations clarify the role of the sampling design. Removing teacher/source diversity, hard-case sampling, or agreement-first target selection reduces aggregate RubricAcc by 3.9–5.5 points relative to Ours-RSFT, with larger 8.1–10.5 point drops in UnsatAcc. These results show that RSFT improves criterion-level faithfulness (R1) not simply by adding more fine-tuning data, but by selecting supervision that is diverse, difficult, and aligned with the target criterion vector.

4.2 Is the Verifier Invariant to Verbosity?

R2 requires the verifier's criterion vector and scalar reward to remain stable under reward-irrelevant surface changes. We test this with paired concise/verbose rewrites whose gold criterion vectors are fixed, so verbose-minus-concise shifts measure surface-form sensitivity.

Figure 4 shows that verbosity raises the base verifier's predicted grade by +5.6 points and false credit by +4.2 points. RubricVerifier-STEM reduces these shifts to +1.5 and +1.6 points, close to GPT-OSS-120B (+2.2 and +1.8), substantially weakening the reward shortcut available to a downstream policy.

4.3 Can the Verifier Serve as an Effective RL Reward?

The preceding sections established that RubricVerifier-STEM satisfies the criterion-vector contract on the static benchmark: criterion-level faithfulness (R1), surface-form invariance (R2), and servable output compliance (R3). We now test whether this contract translates into an effective RL reward and whether the same properties hold under the live rollout workload.

Setup. All configurations start from the same Qwen3-4B-Thinking policy and share the same 8,229-prompt training mixture, GSPO launcher, rollout budget, and 17-set evaluation suite; only the reward source changes. The mixture combines PHYSICS problems (Zheng et al., 2025a), online mathematical-competition material, and DeepMath-103K (He et al., 2025). Rubric-based rewards scalarize criterion verdicts using Equation 1; scalar-judge and outcome baselines predict a scalar reward directly from the same problem and solution. We report Avg17, the unweighted mean of normalized scores over 17 held-out evaluation sets.

Policy gain. Table 4 reports the matched downstream RL comparison. With the policy initialization, data, optimizer, rollout budget, and evaluation held fixed, RubricVerifier-STEM reaches 47.5 Avg17, a +9.3 point gain over the unaugmented baseline (38.3). This is within 0.4 points of the GPT-OSS-120B verifier reward (+9.7) despite a 30× smaller verifier, and +6.2 points above the same-size Qwen3-4B-Instruct reward (+3.0).

Live-rollout contract checks. A reward that performs well offline could still fail in loop: it may be too slow or malformed for rollout-time serving, drift toward verbosity under varied policy samples, or award unjustified high credit when consumed by the optimizer. We therefore check R3, R2, and R1 on the live RL run above. For servable output compliance (R3), Ours-RL reaches

Table 2: Core results and compact reference context on RUBRIC-HARD-STEM. Lower GradeMAE and output tokens are better. All metrics except output tokens are reported as percentages.

Model	RUBRICACC ↑	CRITACC ↑	GradeMAE ↓	F1 ↑	Exact ↑	#Tokens ↓
Closed-source reference judges						
GPT-5.4†	84.9	86.5	12.0	84.3	33.7	62.6
Gemini-3-Flash-Preview	84.5	87.0	13.3	85.7	37.3	3652.1
Open-weight reference models						
GPT-OSS-120B	83.9	87.3	13.8	85.3	35.4	547.8
GPT-OSS-20B	81.9	84.7	15.3	82.3	31.4	650.7
Nemotron-3-Super-120B-A12B	82.6	84.2	15.1	83.6	31.7	1276.0
Qwen3.5-397B-A17B	78.8	80.0	16.3	79.5	28.3	5307.8
Qwen3.5-27B	77.7	80.9	19.1	80.0	25.0	3013.6
Qwen3.5-4B	75.2	78.8	19.1	81.1	21.3	4345.7
RubricVerifier checkpoints						
Qwen3-4B-Instruct (Base)	66.6	68.5	27.5	67.6	11.0	314.8
OURS-RSFT	81.3	84.4	15.8	81.8	29.2	390.7
OURS-RL(RUBRICVERIFIER-STEM)	82.9	85.6	14.9	83.2	31.4	458.8

† Reasoning-token counts are not exposed by the service API; #Tokens reports visible output tokens only.

Table 3: Surface-form and serving diagnostics. Δ columns are verbose-minus-concise shifts in percentage points. Panel (b) reports RSFT ablations relative to complete Ours-RSFT.

(a) Accuracy, serving, and verbosity shift

System	RUBRICACC	FALSECREDIT	Items/s	ΔGrade	ΔFC
Base	66.6	35.4	1.09	+5.6	+4.2
Ours-RL	82.9	10.3	**1.87**	**+1.5**	**+1.6**
GPT-OSS-120B	**83.9**	**9.6**	1.19	+2.2	+1.8

(b) Missing RSFT safeguard

Removed	ΔAcc	ΔUnsat
Teacher/source diversity	-4.2	-8.1
Hard-case sampling	-4.6	-9.9
Agreement-first target	-5.5	-10.5

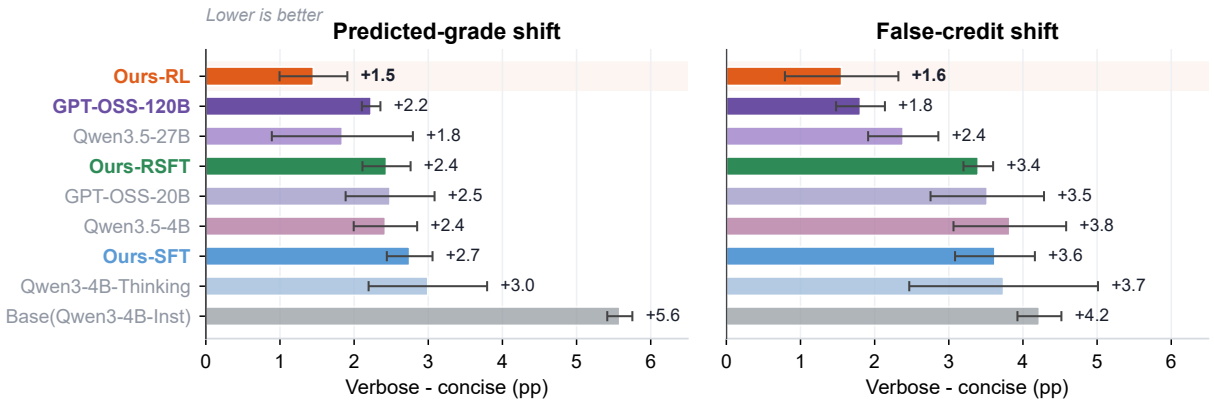

Figure 4: Paired concise/verbose diagnostic with the gold criterion vector held fixed: shift in predicted rubric grade (left) and false-credit rate (right), verbose minus concise (percentage points). Rows are sorted from smallest to largest combined shift, and lower values indicate less sensitivity to superficial padding.

1.87 items/s with a 0.04% parse-fail rate, faster than GPT-OSS-120B (1.19 items/s) and more reliable than the base verifier (1.09 items/s, 0.13% parse-fail), with rollout-time outputs roughly half as long as the base verifier's outputs (304 vs. 797 tokens). For surface-form invariance (R2), we compare response pairs from the same prompt and ask whether the longer response receives the higher score. On rollout samples, this happens less often under OURS-RL than under the base verifier, 42.3% versus 46.3%, suggesting reduced length sensitivity. For criterion-level faithfulness (R1), we replay rollout requests from iterations 181–190 through both verifiers, yielding 304 overlapping iter-prompt groups. Blinded human review judges the higher-reward decision as an unjustified false positive in 32.2% of all overlapping groups for the base verifier, versus 16.8% for OURS-RL, nearly halving in-loop over-crediting. Appendix D.2 further shows that these flagged rewards are not bor-

Table 4: Matched downstream RL outcomes. All rows have 17/17 evaluation coverage and change only the reward source. Avg17 is reported as a percentage; Δ is in percentage points.

Reward source	Granularity	Avg17 (%)	Δ pp
Baseline	none	38.3	+0.0
Math-Verify	answer/outcome	40.6	+2.3
GPT-OSS-120B	outcome scalar	43.7	+5.5
Qwen3-4B-Instruct	criterion vector	41.3	+3.0
Ours-RL	criterion vector	47.5	+9.3
GPT-OSS-120B	criterion vector	**47.9**	**+9.7**

Table 5: **Out-of-diagnostic transfer.** All rows are evaluated with the same parser on a separate 2,410-example suite with 16,746 criteria. Accuracy and MAE are reported as percentages.

Model	#Tokens ↓	ProfBench		RubricBench	
		Acc↑	MAE↓	Acc↑	MAE↓
Gemini-3-Flash	3933.8	74.7	18.7	**89.2**	**9.6**
Qwen3-4B-Inst	307.7	71.0	19.5	77.7	19.0
Qwen3.5-27B	2250.6	74.7	19.5	87.0	11.7
GPT-OSS-120B	356.6	76.6	16.9	88.9	9.9
OURS-RL	413.9	**76.8**	**16.7**	84.4	13.5

derline cases. Overall, OURS-RL maintains the reward-interface contract in loop beyond static tests.

4.4 How Far Does the Verifier Generalize?

We further test whether the learned reward interface remains reliable on broader benchmark suites with larger sample sizes. Table 5 evaluates the same parser and metrics on RubricBench (Zhang et al., 2026) and ProfBench(Wang et al., 2025). OURS-RL preserves the compact serving format while matching or exceeding the strongest baselines on STEM-oriented tasks and improving on general-domain tasks as well. It reaches 76.8% macro RUBRICACC on ProfBench, slightly above GPT-OSS-120B and 4.1 points above Qwen3-4B-Instruct, while also outperforming Qwen3-4B-Instruct on RubricBench. These results provide evidence that verifier-oriented training can transfer beyond the diagnostic benchmark to external rubric-verification settings.

5 Related Work

LLM verifiers and verifier benchmarks. Verifier-based training has become a standard way to turn model outputs into reward signals or diagnostic judgments. Outcome verifiers and RLVR systems judge final-answer correctness or task-level validity (He et al., 2025; Liu et al., 2025; Zheng et al., 2025b; Chen et al., 2025; Yan et al., 2025), while process-supervision methods evaluate intermediate reasoning steps or reasoning traces (Luo et al., 2025; Lu et al., 2024; Zheng et al., 2024). These works show that verifiers can provide useful supervision for reasoning models, and recent benchmarks make verifier failures more measurable. However, their main audit unit is usually final answer or reasoning steps, leaving space for rubric-level audit in RLVR.

LLM judges, rubric grading, and rubric rewards. A parallel line of work uses LLMs as judges, often with criteria or rubrics, to produce scalar scores, pairwise preferences, critiques, or grading decisions (Liu et al., 2023; Zheng et al., 2023; Fu et al., 2024; Kocmi and Federmann, 2023; Kim et al., 2024; Zhu et al., 2025; Hashemi et al., 2024; Deng et al., 2026; Pathak et al., 2025; Eltanbouly et al., 2025; Rao and Callison-Burch, 2026; Wang et al., 2025). This literature also documents judge biases such as position, style, and length effects, which become especially problematic when judge outputs are used as rewards (Shi et al., 2024; Ye et al., 2024; Dubois et al., 2024). These studies establish that rubrics can make open-ended supervision more structured, but comparatively less work isolates the rubric-conditioned verifier itself as the reliability bottleneck in RL reward serving. We study this missing layer by formulating a criterion-vector reward contract.

6 Conclusion

Rubrics offer a natural source of dense supervision for STEM reasoning, but they are not rewards by themselves: they become useful for RL only through a verifier interface that assigns credit faithfully, resists surface-form shortcuts, and produces outputs that can be served reliably during rollout. This paper formalizes rubric-conditioned grading as a criterion-vector reward interface and evaluates that interface through a three-part reward-interface contract. Empirically, RUBRICVERIFIER-STEM satisfies this contract more reliably than the base verifier while remaining compact. When scalarized, its criterion-level verdicts also transfer effectively as reward signals for matched downstream RL. We hope this work moves rubric-based RL toward a more systematic and effective paradigm by turning rubrics from informal grading specifications into explicit, testable, and servable reward interfaces.

Limitations

Criterion-level auditing depends on rubric quality. If the rubric omits an important requirement, decomposes the task poorly, or encodes a contestable grading policy, the verifier can only expose that flawed decomposition faithfully. Better local verdicts do not by themselves resolve ambiguity in the rubric.

RUBRIC-HARD-STEM has only 300 examples and focuses on long-form STEM/applied-engineering grading. It does not cover handwritten work, diagrams, multimodal solutions, or rubrics that require ordinal, dependent, uncertain, or continuous partial credit. Transfer to other domains, other rubric styles, or weaker rubrics remains uncertain.

The downstream validation uses one 4B thinking-policy initialization, one 8,229-prompt PHYSICS/-math training mixture, and one 17-set evaluation suite. Time and compute constraints prevented broader sweeps over larger policy backbones, reward models, and data mixtures. This scopes the downstream claim to practical usefulness under a matched setup.

Ethical Considerations

A criterion-level verdict is only as fair as the rubric it operationalizes. While criterion-level outputs can improve auditability, they may also inherit or amplify contestable rubric assumptions and appear more objective than scalar grades even when the rubric encodes debatable judgments. We therefore view RUBRICVERIFIER-STEM as decision support and training feedback, not as a replacement for human review in high-stakes grading. All annotators were compensated fairly and in accordance with applicable regulations and institutional guidelines. We use all source datasets and model artifacts under their stated licenses, access conditions, and research-use terms; derived artifacts are intended for research use only, and restricted source content is not redistributed outside permitted conditions. The data consist of STEM problems, rubrics, generated solutions, and criterion-level labels; we do not intentionally collect personal data, and any public release will be screened for personally identifying information and offensive content. Dataset releases should document provenance, annotation policy, known biases, and licensing, and deployments should include appeal paths, hidden validation splits, and red-team tests for verbosity, adversarial formatting, rubric copying, and domain-specific bias.

References

Ding Chen, Qingchen Yu, Pengyuan Wang, Wentao Zhang, Bo Tang, Feiyu Xiong, Xinchi Li, Minchuan Yang, and Zhiyu Li. xVerify: Efficient answer verifier for reasoning model evaluations. *arXiv preprint arXiv:2504.10481*, 2025. doi: 10.48550/arXiv.2504.10481. URL https://arxiv.org/abs/2504.10481.

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

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. doi: 10.48550/arXiv.2501.12948. URL https://arxiv.org/abs/2501.12948.

Haotian Deng, Chris Farber, Jiyoon Lee, and David Tang. Rubric-conditioned LLM grading: Alignment, uncertainty, and robustness. arXiv preprint arXiv:2601.08843, 2026. URL https://arxiv.org/abs/2601.08843.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. Length-controlled AlpacaEval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024. doi: 10.48550/arXiv.2404.04475. URL https://arxiv.org/abs/2404.04475.

Sohaila Eltanbouly, Salam Albatarni, and Tamer Elsayed. TRATES: Trait-specific rubric-assisted cross-prompt essay scoring. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20528–20543, Vienna, Austria, jul 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.1054. URL https://aclanthology.org/2025.findings-acl.1054/.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. GPTScore: Evaluate as you desire. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6556–6576, Mexico City, Mexico, jun 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.365. URL https://aclanthology.org/2024.naacl-long.365/.

Google DeepMind. Gemma 4 model card. https://ai.google.dev/gemma/docs/core/model_card_4, April 2026. Accessed: 2026-05-25.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. arXiv preprint arXiv:2507.17746, 2025. URL https://arxiv.org/abs/2507.17746.

Helia Hashemi, Jason Eisner, Corby Rosset, Benjamin Van Durme, and Chris Kedzie. LLM-rubric: A multidimensional, calibrated approach to automated evaluation of natural language texts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13806–13834, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.745. URL https://aclanthology.org/2024.acl-long.745/.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. DeepMath-103K: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025. doi: 10.48550/arXiv.2504.11456. URL https://arxiv.org/abs/2504.11456.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Jiaqi Hu, Zhanming Shen, Xiaomeng Hu, Xijun Gu, Peiyi Tu, Jiaxin Liu, Wenyu Chen, Yuzhuo Fu, Zhiting Fan, Yanmei Gu, Yuanyuan Wang, Zhengkai Yang, Jianguo Li, and Junbo Zhao. Reinforcement learning with rubric anchors, 2025. URL https://arxiv.org/abs/2508.12790.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4334–4353, Miami, Florida, USA, nov 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.248. URL https://aclanthology.org/2024.emnlp-main.248/.

Tom Kocmi and Christian Federmann. Large language models are state-of-the-art evaluators of translation quality. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 193–203, Tampere, Finland, jun 2023. European Association for Machine Translation. URL https://aclanthology.org/2023.eamt-1.19/.

Sunzhu Li, Jiale Zhao, Miteto Wei, Huimin Ren, Yang Zhou, Jingwen Yang, Shunyu Liu, Kaike Zhang, and Wei Chen. RubricHub: A comprehensive and highly discriminative rubric dataset via automated coarse-to-fine generation. arXiv preprint arXiv:2601.08430, 2026. URL https://arxiv.org/abs/2601.08430.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Shudong Liu, Hongwei Liu, Junnan Liu, Linchen Xiao, Songyang Gao, Chengqi Lyu, Yuzhe Gu, Wenwei Zhang, Derek F. Wong, Songyang Zhang, and Kai Chen. CompassVerifier: A unified and robust verifier for LLMs evaluation and outcome reward. *arXiv preprint arXiv:2508.03686*, 2025. doi: 10.48550/arXiv.2508.03686. URL https://arxiv.org/abs/2508.03686.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore, dec 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Jianqiao Lu, Zhiyang Dou, Hongru Wang, Zeyu Cao, Jianbo Dai, Yingjia Wan, and Zhijiang Guo. AutoPSV: Automated process-supervised verifier. *arXiv preprint arXiv:2405.16802*, 2024. doi: 10.48550/arXiv.2405.16802. URL https://arxiv.org/abs/2405.16802.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models with automated process supervision. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=KwPUQOQIKt.

Thang Luong, Dawsen Hwang, Hoang H. Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu Hoang Trinh, Quoc V. Le, and Junehyuk Jung. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35418–35442, Suzhou, China, nov 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1794. URL https://aclanthology.org/2025.emnlp-main.1794/.

OpenAI, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Aditya Pathak, Rachit Gandhi, Vaibhav Uttam, Devansh, Yashwanth Nakka, Aaryan Raj Jindal, Pratyush Ghosh, Arnav Ramamoorthy, Shreyash Verma, Aditya Mittal, Aashna Ased, Chirag Khatri, Jagat Sesh Challa, and Dhruv Kumar. Rubric is all you need: Enhancing LLM-based code evaluation with question-specific rubrics. arXiv preprint arXiv:2503.23989, 2025. URL https://arxiv.org/abs/2503.23989. Accepted to ICER 2025.

Qwen Team, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

Delip Rao and Chris Callison-Burch. Autorubric: Unifying rubric-based LLM evaluation. arXiv preprint arXiv:2603.00077, 2026. URL https://arxiv.org/abs/2603.00077.

Lin Shi, Chiyu Ma, Wenhua Liang, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic investigation of position bias in pairwise comparative assessments by llms. OpenReview preprint, 2024. URL https://openreview.net/forum?id=y3jJmrKWQ4.

Miles Wang, Robi Lin, Kat Hu, Joy Jiao, Neil Chowdhury, Ethan Chang, and Tejal Patwardhan. FrontierScience: Evaluating AI's ability to perform expert-level scientific tasks. *arXiv preprint arXiv:2601.21165*, 2026. doi: 10.48550/arXiv.2601.21165. URL https://arxiv.org/abs/2601.21165.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand, aug 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Zhilin Wang, Jaehun Jung, Ximing Lu, Shizhe Diao, Ellie Evans, Jiaqi Zeng, Pavlo Molchanov, Yejin Choi, Jan Kautz, and Yi Dong. ProfBench: Multi-domain rubrics requiring professional knowledge to answer and judge. *arXiv preprint arXiv:2510.18941*, 2025. doi: 10.48550/arXiv.2510.18941. URL https://arxiv.org/abs/2510.18941.

Yuchen Yan, Jin Jiang, Zhenbang Ren, Yijun Li, Xudong Cai, Yang Liu, Xin Xu, Mengdi Zhang, Jian Shao, Yongliang Shen, Jun Xiao, and Yueting Zhuang. VerifyBench: Benchmarking reference-based reward systems for large language models. *arXiv preprint arXiv:2505.15801*, 2025. doi: 10.48550/arXiv.2505.15801. URL https://arxiv.org/abs/2505.15801.

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, Nitesh V. Chawla, and Xiangliang Zhang. Justice or prejudice? quantifying biases in LLM-as-a-judge. *arXiv preprint arXiv:2410.02736*, 2024. doi: 10.48550/arXiv.2410.02736. URL https://arxiv.org/abs/2410.02736.

Fangchen Yu, Haiyuan Wan, Qianjia Cheng, Yuchen Zhang, Jiacheng Chen, Fujun Han, Yulun Wu, Junchi Yao, Ruilizhen Hu, Ning Ding, Yu Cheng, Tao Chen, Lei Bai, Dongzhan Zhou, Yun Luo, Ganqu Cui, and Peng Ye. HiPhO: How far are (M)LLMs from humans in the latest high school physics olympiad benchmark? *arXiv preprint arXiv:2509.07894*, 2025. doi: 10.48550/arXiv.2509.07894. URL https://arxiv.org/abs/2509.07894.

Qiyuan Zhang, Junyi Zhou, Yufei Wang, Fuyuan Lyu, Yidong Ming, Can Xu, Qingfeng Sun, Kai Zheng, Peng Kang, Xue Liu, and Chen Ma. RubricBench: Aligning model-generated rubrics with human standards. *arXiv preprint arXiv:2603.01562*, 2026. doi: 10.48550/arXiv.2603.01562. URL https://arxiv.org/abs/2603.01562.

Zehua Zhao, Zhixian Huang, Junren Li, Siyu Lin, Junting Zhou, Fengqi Cao, Kun Zhou, Rui Ge, Tingting Long, Yuexiang Zhu, Yan Liu, Jie Zheng, Junnian Wei, Rong Zhu, Peng Zou, Wenyu Li, Zekai Cheng, Tian Ding, Yaxuan Wang, Yizhao Yan, Tingru Wei, Haowei Ming, Weijie Mao, Chen Sun, Yiming Liu, Zichen Wang, Zuo Zhang, Tong Yang, Hao Ma, Zhen Gao, and Jian Pei. SUPERChem: A multimodal reasoning benchmark in chemistry. *arXiv preprint arXiv:2512.01274*, 2025. doi: 10.48550/arXiv.2512.01274. URL https://arxiv.org/abs/2512.01274.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*, 2024. doi: 10.48550/arXiv.2412.06559. URL https://arxiv.org/abs/2412.06559.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Shenghe Zheng, Qianjia Cheng, Junchi Yao, Mengsong Wu, Haonan He, Ning Ding, Yu Cheng, Shuyue Hu, Lei Bai, Dongzhan Zhou, Ganqu Cui, and Peng Ye. Scaling physical reasoning with the PHYSICS dataset. In *Advances in Neural Information Processing Systems 38 (Datasets and Benchmarks Track)*, 2025a. doi: 10.48550/arXiv.2506.00022. URL https://arxiv.org/abs/2506.00022.

Shenghe Zheng, Chenyu Huang, Fangchen Yu, Junchi Yao, Jingqi Ye, Tao Chen, Yun Luo, Ning Ding, Lei Bai, Ganqu Cui, and Peng Ye. SCI-Verifier: Scientific verifier with thinking. *arXiv preprint arXiv:2509.24285*, 2025b. doi: 10.48550/arXiv.2509.24285. URL https://arxiv.org/abs/2509.24285.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. JudgeLM: Fine-tuned large language models are scalable judges. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=87YOFayjcG.

Appendix

A Benchmark and Reward Interface

A.1 Sampling, Scope, and Composition

Gold grades are stratified into six 50-example regimes: one endpoint regime pooling exact 0 and exact 1 grades, and five interior regimes covering $(0, 0.2)$, $[0.2, 0.4)$, $[0.4, 0.6)$, $[0.6, 0.8)$, and $[0.8, 1.0)$. The endpoint regime is used only for sampling coverage; all reported metrics use the original criterion labels and derived grades. The engineering bucket includes code and software-engineering cases plus a smaller number of other applied- engineering prompts that satisfy the same rubric-quality filters.

Table 6: Reward-interface comparison. Criterion-level verification preserves scalar aggregation while exposing local credit decisions hidden by answer-level and scalar interfaces.

Interface	Output	Local reward failure exposed
Answer verifier	final-answer score	valid partial credit lost after final-answer errors
Scalar judge / RM	score or preference	false credit on unsatisfied criteria
Process verifier	step-level scores	step correctness may not match rubric clauses
Rubric verifier	**criterion vector**	**rubric-defined credit units are explicit**

Table 7: Composition of RUBRIC-HARD-STEM. Satisfied and unsatisfied counts are criterion labels, not examples.

Domain	Ex.	Criteria	Sat.	Unsat.
Math	60	243	120	123
Physics	60	939	377	562
Chemistry	60	614	320	294
Biology	60	773	243	530
Engineering	60	380	244	136
Total	**300**	**2,949**	**1,304**	**1,645**

A.2 Pipeline Safeguards

Table 8 expands the safeguards summarized by Figure 2. The main design point is that the reward interface is hardened against both semantic errors (wrong local verdicts) and serving errors (missing, extra, or non-parseable verdicts).

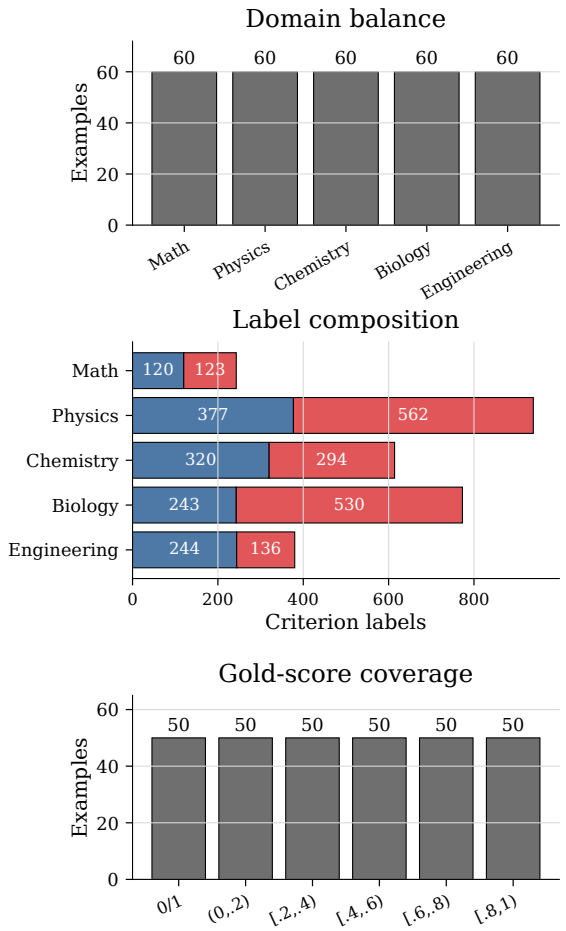

Figure 5: RUBRIC-HARD-STEM balance controls before hard-case mining: domain coverage, criterion-label composition, and gold-score coverage.

Table 8: Reward risks and pipeline safeguards used in the training recipe in Figure 2.

Risk	Pipeline safeguard	Diagnostic
Lenient–strict calibration	Balance domains and gold-score bins before hard-case mining; include satisfied, unsatisfied, and high-disagreement cases.	RUBRICACC, SATACC, UNSATACC, FALSECREDIT
Accuracy–length tradeoff	Use strong offline judgments for labels, then keep the shortest parser-valid target among outputs with near-best criterion-vector agreement.	RUBRICACC, tokens, latency
Verbosity/style shortcuts	Train with length pairs and hard negatives; optimize parse validity, coverage, and false-credit-aware rewards during GSPO.	Concise/verbose pairs; parse failures

B Additional Diagnostics

Table 9: Serving and surface-form diagnostics. Serving metrics are measured on 1,024 saved rollout reward requests. Δ columns are verbose-minus-concise shifts in percentage points. Dashes indicate systems not evaluated in that diagnostic.

	Reward-server serving			Concise/verbose paired diagnostic			
Model	Items/s	Med.	p95	ΔGrade	ΔFC	V>C	C>V
Qwen3-4B-Instruct	1.09	10.44	37.91	+5.6	+4.2	30.6	13.9
Qwen3-4B-Thinking	0.41	37.78	79.51	+3.0	+3.7	29.3	12.8
OURS-SFT	–	–	–	+2.7	+3.6	30.7	16.7
OURS-RSFT	–	–	–	+2.4	+3.4	27.1	15.9
OURS-RL	**1.87**	**6.45**	**20.16**	**+1.5**	**+1.6**	26.1	16.2
GPT-OSS-120B	1.19	13.40	20.30	+2.2	+1.8	25.2	13.0
Qwen3.5-27B	–	–	–	+1.8	+2.4	26.9	19.6

Table 10: Expanded reference tier with open-weight rows and service-reported remote judges. Exact is the proportion of examples where every predicted verdict agrees with the ground-truth verdict. Lower GradeMAE and output tokens are better. All metrics except output tokens are reported as percentages.

Model	RUBRICACC	CRITACC	GradeMAE	F1	Exact	output tokens
Service-based contextual judges						
gpt-5.4	**84.9**	86.5	12.0	84.3	33.7	62.6
gemini-3-flash-preview	84.5	87.0	13.3	85.7	37.3	3652.1
Open-weight and trained verifier rows						
GPT-OSS-120B	83.9	87.3	13.8	85.3	35.4	547.8
OURS-RL	82.9	85.6	14.9	83.2	31.4	458.8
Nemotron-3-Super-120B-A12B	82.6	84.2	15.1	83.6	31.7	1276.0
Gemma4-distill-SFT-DPO	81.9	84.7	15.6	82.4	29.9	437.0
GPT-OSS-20B	81.9	84.7	15.3	82.3	31.4	650.7
OURS-RSFT	81.3	84.4	15.8	81.8	29.2	390.7
Qwen3.5-397B-A17B	78.8	80.0	16.3	79.5	28.3	5307.8
Qwen3.5-27B	77.7	80.9	19.1	80.0	25.0	3013.6
Qwen3.5-4B	75.2	78.8	19.1	81.1	21.3	4345.7
OURS-SFT	76.2	76.5	19.4	75.8	21.9	323.8
Qwen3-4B-Thinking	71.9	76.3	23.0	77.0	19.4	2718.2
Qwen3-4B-Instruct	66.6	68.5	27.5	67.6	11.0	314.8

Table 11: Checkpoint and RSFT-stage ablations on RUBRIC-HARD-STEM. Ablations remove one pre-GSPO safeguard and should be compared with the complete OURS-RSFT row; OURS-RL is included only as the final GSPO reference. All metrics are percentages.

Variant	Source or contrast	RUBRICACC	CRITACC	GradeMAE	BalAcc	SATACC	UNSATACC	Miss. rate
Reference checkpoints								
Qwen3-4B-Instruct	none	66.6	68.5	27.5	69.0	73.5	64.6	0.3
OURS-SFT	diverse-source SFT	76.2	76.5	19.4	77.1	82.2	72.0	0.5
OURS-RL	diverse SFT + RSFT + GSPO	**82.9**	**85.6**	**14.9**	**85.1**	80.6	**89.7**	**0.0**
RSFT-stage ablations; compare against complete OURS-RSFT								
OURS-RSFT	diverse SFT + RSFT	81.3	84.4	15.8	83.8	78.8	88.8	0.1
RSFT ablation	single teacher family	77.1	78.6	20.6	78.4	76.1	80.7	0.1
RSFT ablation	without hard negatives	76.7	78.4	20.3	78.3	77.7	78.9	1.1
RSFT ablation	without length pairs	77.4	78.6	19.6	78.6	78.9	78.4	0.5
RSFT ablation	non-agreement-first target	75.8	77.2	21.4	77.2	76.1	78.3	0.3

C Training and Reward Implementation

C.1 GSPO Objective

For prompt p, sampled verifier outputs $\{o_i\}_{i=1}^{G}$, and old policy $\pi_{\theta_{\text{old}}}$, GSPO uses the sequence-level importance ratio

$$s_i(\theta) = \exp\left\{ \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \Big[\log \pi_\theta(o_{i,t} \mid p, o_{i,<t}) - \log \pi_{\theta_{\text{old}}}(o_{i,t} \mid p, o_{i,<t}) \Big] \right\}. \tag{2}$$

The clipped objective is

$$\bar{s}_i(\theta) = \text{clip}\big(s_i(\theta), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}\big),$$
$$\mathcal{J}_{\text{GSPO}}(\theta) = \mathbb{E}\left[\frac{1}{G} \sum_{i=1}^{G} \min\big(s_i(\theta)\hat{A}_i, \bar{s}_i(\theta)\hat{A}_i\big) \right], \tag{3}$$

with $G = 8$, $\epsilon_{\text{low}} = 0.2$, and $\epsilon_{\text{high}} = 0.28$. Advantages are group-centered without standard-deviation normalization:

$$\widehat{A}_i = \tilde{r}_i - \frac{1}{G} \sum_{j=1}^{G} \tilde{r}_j. \tag{4}$$

C.2 Verifier Reward and Rollout Mining

The verifier rollout reward is defined over the parsed criterion vector produced at serving time. Correct criterion-level verdicts provide the primary positive signal; incorrect judgments on perturbed training samples, especially those rarely handled correctly by the OURS-RSFT checkpoint and likely related to length preference, incur additional penalties. Malformed, missing, duplicated, or extraneous verdicts are penalized, so parse validity and criterion coverage are hard requirements for reward computation.

To focus GSPO on difficult states, each sampled group receives an EMA-smoothed `score_norm`; we keep groups with `score_norm` in $[0.05, 0.95]$ and at least two samples. This concentrates updates on groups where the verifier still oscillates between correct and incorrect verdict vectors.

All experiments are carried out on 8 NVIDIA H200 GPUs unless otherwise stated. Tables 12–14 summarize the three training stages. SFT/RSFT train the verifier interface, verifier GSPO further optimizes the verifier into OURS-RL, and downstream GSPO freezes the trained verifier as a remote reward model to train a separate reasoning policy.

Table 12: SFT/RSFT settings for the verifier. Both stages train the criterion-vector verifier interface, with RSFT selecting concise targets only after verdict faithfulness is satisfied.

Aspect	Key setting
Epochs	2 epochs (Both SFT/RSFT)
Hardware/layout	1 node; 8 GPUs by default, with 1 GPU reserved for online evaluation
Batching	Training batch size is set to the number of training GPUs $\times 4$
Optimizer	Adam; LR 8.5×10^{-6}; minimum LR 8.5×10^{-7}; cosine decay; warmup fraction 0.1; weight decay 0.1; $\beta_1 = 0.9, \beta_2 = 0.95$
Sequence limits	Rollout context 32,768; prompt cap 32,767; max tokens/GPU 8,192

Table 13: Verifier GSPO settings behind the OURS-RL checkpoint.

Aspect	Key setting
Rollouts	batch 64; 8 samples per prompt; max response 8,192; temperature 1.0
Objective	GSPO advantages; TIS enabled; reward normalization disabled; KL and entropy coefficients set to 0
Clipping	$\epsilon_{\text{low}} = 0.2, \epsilon_{\text{high}} = 0.28$
Optimizer	Adam; LR 1×10^{-7}; cosine decay; weight decay 0.1; global batch 64
Reward terms	parse bonus 0.2; exact bonus 1.0; extra marked-verdict penalty 1.0; reward clipped to $[-2, 2]$
Difficulty gate	EMA $\alpha = 0.3$; keep groups with `score_norm` in $[0.05, 0.95]$ and at least two samples

Table 14: Downstream GSPO policy-training settings. The trained verifier is frozen as a remote reward model and used to train a separate reasoning policy.

Aspect	Key setting
Sampling	65,536-token context; max response 32,768; rollout batch 64; 16 samples/prompt; temperature 1.0
Objective	GSPO with TIS; $\epsilon_{\text{low}} = 0.2$, $\epsilon_{\text{high}} = 0.28$; KL/entropy coefficients 0
Optimizer	Adam; LR 1×10^{-6}; constant LR; weight decay 0.1; $\beta_1 = 0.9, \beta_2 = 0.98$
Training/eval	200 training iterations; save/eval every 20 iterations

C.3 Training Recipe Checklist

The supervised stages use Qwen3-4B-Instruct as initialization and exclude RUBRIC-HARD-STEM from training and checkpoint selection. SFT teaches the task format and broad partial-credit coverage. RSFT selects concise targets only af-

ter criterion-vector agreement is satisfied. Downstream policy experiments initialize from Qwen3-4B-Thinking and keep the prompt mixture, rollout budget, optimizer, and 17-set evaluation fixed across reward sources.

D Evaluation and Audit Protocol

D.1 Expert Reliability and Metric Definitions

Automated verifier scores and deliberative judge passes are used only to prioritize examples for review, never to assign labels. Final criterion labels are produced by blinded experts who inspect the problem, reference solution, rubric, candidate answer, and criterion under review. Disagreements are resolved by a senior expert who is also blinded to answer-generator and verifier metadata. The audit is criterion-level rather than answer-level: annotators decide one rubric clause at a time, so a correct scalar total cannot hide swapped local credit.

D.2 Rollout Disagreement Audit

This audit checks whether false-credit behavior measured on the static benchmark also appears when the verifier is used as a rollout-time reward model. We replay requests from iterations 181–190 of the matched RL run in Section 4.3. For each replay seed and each $(\text{iter}, \text{prompt_hash})$ group, per-sample rewards are averaged to obtain s_{ours} and s_{base}, yielding 304 overlapping run-specific groups.

A group is assigned to the base-higher direction if $s_{\text{base}} > s_{\text{ours}}$ and to the ours-higher direction if $s_{\text{ours}} > s_{\text{base}}$; ties are reported but not audited. Each high-score disagreement is audited by an external auxiliary judge, blinded to verifier identity and disagreement direction, who sees the problem, rubric, and representative high-score response and returns a justified/false-positive/ uncertain verdict plus a corrected scalar grade. For each iteration–prompt group, we compare the reward assigned by the base verifier and by OURS-RL. The Direction column indicates which verifier assigns the higher reward. We then ask a blinded human auditor whether this higher reward is justified by the rubric and the sampled solution; if not, the case is counted as an FP, i.e., an unsupported high reward that could mislead the optimizer.

Among the 304 overlapping groups, the base verifier assigns a higher reward than OURS-RL in 126 cases. In 98 of these cases, or 77.8% of this direction, the higher base reward is judged unsupported, accounting for 32.2% of all audited groups. By contrast, OURS-RL assigns a higher reward than the base verifier in 91 cases, of which 51 are judged unsupported, accounting for 16.8% of all audited groups. Thus, relative to the base verifier, OURS-RL nearly halves the rate of optimizer-facing false positives on the shared rollout workload. The Gap column further shows that the unsupported reward gaps are smaller for OURS-RL-higher cases than for base-higher cases, suggesting that OURS-RL reduces not only the frequency but also the magnitude of over-crediting.

The trained criterion-level verifier does not eliminate over-crediting, but it makes high rewards less often contradicted by the in-loop audit. This complements the R3 serving result: OURS-RL is faster and more parseable at rollout time, while also producing fewer exploitable high-reward mistakes than the same-size instruction-model verifier.

Table 15: Metric definitions used for criterion-vector reward evaluation. Missing, duplicate, extra, or unparsable verdicts are counted as incorrect criterion decisions and as interface failures.

Metric	Meaning
RUBRICACC	Weighted fraction of correctly predicted criteria within each example, macro-averaged across examples.
CRITACC	Micro accuracy over all criterion labels.
GradeMAE	Mean absolute difference between scalarized predicted grade and scalarized gold grade.
SATACC / UNSATACC	Accuracy on gold-satisfied and gold-unsatisfied criteria.
FALSECREDIT	Fraction of gold-unsatisfied criteria incorrectly marked satisfied.
FALSENEGATIVE	Fraction of gold-satisfied criteria incorrectly rejected.
Parser failures	Missing, duplicated, extra, or malformed verdicts in the reward output.

Table 16: In-loop reward disagreement audit. Groups are formed over iterations 181–190. FP means that the higher reward is judged unjustified.

Direction	Groups	FP	FP/all	Reward Gap
Base > OURS-RL	126	98 (77.8%)	32.2%	0.560
OURS-RL > Base	91	51 (56.0%)	16.8%	0.353
Tie	87	–	–	–
Total	304	–	–	–

E Prompt Template

E.1 Parser-Valid Verifier Interface Template

Input blocks. The reward server supplies: (i) the problem, (ii) the candidate/student response, and (iii) an instance-specific rubric with criteria indexed from 1 to m and optional weights.

Judgment contract. The verifier may construct a

short internal analysis, but the parser-facing output must contain exactly one verdict for every rubric criterion and no extra criterion ids.

```
Criterion 1: Satisfied | Not Satisfied
Criterion 2: Satisfied | Not Satisfied
...
Criterion m: Satisfied | Not Satisfied
```

Missing, duplicated, extra, or malformed verdicts are counted as incorrect and as parser-interface failures; scalar reward is computed only after the criterion vector is parsed.

Rubric verification prompt specification

Role You are an expert grader. Given a problem, a student solution, and K marking criteria, evaluate the solution against each criterion independently.

Task For each criterion $n \in \{1, \ldots, K\}$, output exactly one line:

```
Criterion <n>:  Satisfied
Criterion <n>:  Not Satisfied
```

You may think step by step inside `<think>...</think>` tags before giving the final verdicts.

Problem: `{problem}`
Student Solution: `{student_solution}`
Marking Criteria:
Criterion 1: `{criterion_1}`
Criterion 2: `{criterion_2}`
...
Criterion K: `{criterion_K}`
Your thinking process and verdicts:

Human Expert Validation of Rewrites. After automatic generation, each concise and verbose rewrite is independently reviewed by human experts. The experts verify whether the rewrite preserves the same final answer, follows the same reasoning trajectory, and remains faithful to the original solution. A rewrite is accepted only if it changes the presentation style without changing the underlying reasoning. Rewrites that alter the solution strategy, introduce new assumptions, omit necessary reasoning steps, or change the semantic content of the answer are rejected.

Concise/Verbose reasoning-preserving rewrite prompt specification

Role You are an expert solution rewriter. Given a problem, an original answer, an observed reasoning trace, a ground-truth final answer, and a target rewrite style, rewrite the answer while preserving its reasoning and final conclusion.

Task Rewrite the original answer according to the target rewrite style `{rewrite_style}`, where the style is either `concise` or `verbose`.
If the target style is `concise`, produce a shorter and more direct version that removes redundancy.
If the target style is `verbose`, produce a more detailed and explanatory version that makes implicit steps more explicit.
The rewrite must satisfy all of the following constraints:

- It must preserve the same final answer as the ground truth.
- It must preserve the same actual reasoning process as the original answer.
- It must not omit any reasoning step necessary for obtaining the ground-truth answer.
- It may remove redundancy for a concise rewrite, but must not remove necessary reasoning.
- It may make implicit steps more explicit for a verbose rewrite, but only when those steps are already implied by the original reasoning.
- It must not introduce new assumptions, new solution strategies, or unsupported intermediate claims.
- It must not correct the solution using reasoning that is absent from the original answer.
- If the original answer cannot be faithfully rewritten while remaining consistent with the ground truth, output `Invalid for Rewrite`.

The output must use the following format exactly:

```
Rewrite Style:
{rewrite_style}
Rewrite:
{rewrite}
Rewrite Validity:  Valid
```

If the answer cannot be faithfully rewritten:

```
Rewrite Style:
{rewrite_style}
Rewrite Validity:  Invalid
for Rewrite
```

Problem: `{problem}`
Original Answer: `{original_answer}`
Observed Reasoning Trace: `{reasoning_trace}`
Ground-Truth Final Answer: `{gt_answer}`
Rewrite Style: `{rewrite_style}`
Your rewrite:

Full Case 1: Math short-answer rubric (missed credit)

Trace metadata for MATH-09

Representative point Comparison judge misses credit on 1 satisfied criterion/criteria.

Parsed vectors Gold `[1,1,1]`; RubricVerifier-STEM `[1,1,1]`; comparator `[1,1,0]`.

Parsed scores RubricVerifier-STEM criterion accuracy 1.00a; comparator criterion accuracy 0.67; false credit 0; missed credit 1.

Raw judge output lengths RubricVerifier-STEM 144 words; comparator 176 words.

Original Problem

```
What is the sum of the lengths, in centimeters,
      of the two legs of a 30-60-90 right
     triangle, if the length of the hypotenuse
     is $2\sqrt{6}$ centimeters?
```

Original Student Answer

```
Here is the same solution written more fully.

In a \(30^\circ\)-\(60^\circ\)-\(90^\circ\)
     right triangle, the side lengths always
     follow the standard ratio

\[
1:\sqrt3:2,
\]

where the shorter leg corresponds to \(1\), the
      longer leg corresponds to \(\sqrt3\), and
     the hypotenuse corresponds to \(2\). So
     once the hypotenuse is known, the other two
      sides can be found directly from that
     ratio.

**1. Let the shorter leg be \(x\)**

If the shorter leg is \(x\), then the
     hypotenuse must be \(2x\). The problem
     tells us that the hypotenuse is \(2\sqrt6\)
     , so

\[
2x=2\sqrt6.
\]

Dividing both sides by \(2\) gives

\[
x=\sqrt6.
\]

So the shorter leg is \(\sqrt6\) centimeters.

**2. Find the longer leg from the ratio**

In the same special triangle, the longer leg is
      \(\sqrt3\) times the shorter leg.
     Therefore,

\[
\text{longer leg}=\sqrt3\cdot \sqrt6.
\]

Multiply the radicals:

\[
\sqrt3\cdot \sqrt6=\sqrt{18}=3\sqrt2.
\]

So the longer leg is \(3\sqrt2\) centimeters.

**3. Add the two legs**

The sum requested in the problem is

\[
\sqrt6+3\sqrt2.
\]

Therefore, the sum of the lengths of the two
     legs is

\[
\boxed{\sqrt6+3\sqrt2\text{ centimeters}}.
\]

So the whole problem is just an application of
     the standard \(30\)-\(60\)-\(90\) side
     ratio.
```

Rubric / Marking Criteria

```
Criterion 1: Does the solution give the sum of
     the two legs as sqrt6 + 3sqrt2?
Criterion 2: Does the solution directly use the
      30-60-90 ratio "hypotenuse = 2 x shorter
     leg" to set up 2x = 2sqrt6 and solve for
     the shorter leg immediately?
Criterion 3: Does the reasoning avoid repeating
      the same information or formulas, instead
     using a compact sequence of steps from the
     side ratios to the sum of the legs?
```

Original Judge Response: RubricVerifier

```
1. Solution Analysis: The student correctly
     applied the 30-60-90 side ratio, set 2x = 2
     sqrt6 to find the shorter leg x = sqrt6,
     computed the longer leg as sqrt3*sqrt6 = 3
     sqrt2, and gave the sum of the two legs as
     sqrt6 + 3sqrt2 cm. The reasoning is concise
      and avoids unnecessary repetition.

2. Criterion-by-Criterion Evaluation:
   - Criterion 1: The final answer is sqrt6 + 3
     sqrt2 cm. -> Satisfied -> 5 pts
   - Criterion 2: The solution directly uses
     the ratio "hypotenuse = 2 x shorter leg" (2
     x = 2sqrt6) and solves for the shorter leg.
      -> Satisfied -> 5 pts
   - Criterion 3: The steps are presented in a
     compact sequence without repeating the same
      information. -> Satisfied -> 4 pts

3. Score Calculation: 5 + 5 + 4 = 14
</think>

Criterion 1: Satisfied
Criterion 2: Satisfied
Criterion 3: Satisfied

Final Score:
\boxed{14}
```

Original Judge Response: GPT-OSS-120B (temp0.7 r2)

```
<|channel|>analysis<|message|>We need to
     evaluate each criterion.

Criterion 1: Does solution give sum as sqrt6 +
     3sqrt2? Yes, they concluded that.

Criterion 2: Does solution directly use the
     ratio hypotenuse = 2 x shorter leg to set
     up 2x = 2sqrt6 and solve? Yes, they did.

Criterion 3: Does reasoning avoid repeating
     same info or formulas, using compact
     sequence? The solution repeats ratio
     description multiple times, but they still
     present steps. The criterion asks to avoid
     repeating same info; they restated ratio
```

```
     initially, then later again "In the same
     special triangle...". Might be considered
     some redundancy. The solution is somewhat
     verbose, but still fairly compact? They
     repeated the ratio explanation, but not
     overly. The criterion likely expects a
     concise solution without redundant
     restatement. The student's solution does
     repeat the ratio twice, but overall still
     uses a sequence. Might be considered not
     satisfied because of repetition. I think
     they did repeat the same information: they
     restated the ratio and the longer leg
     relation. So Not Satisfied.

Thus final lines.<|end|><|start|>assistant<|
     channel|>final<|message|>Criterion 1:
     Satisfied
Criterion 2: Satisfied
Criterion 3: Not Satisfied
```

Full Case 2: Chemistry production-rate rubric (false credit)

Trace metadata for CHEMISTRY-05

Representative point Comparison judge gives false credit on 1 unsatisfied criterion/criteria.

Parsed vectors Gold `[1,1,1,0]`; RubricVerifier-STEM `[1,1,1,0]`; comparator `[1,1,1,1]`.

Parsed scores RubricVerifier-STEM criterion accuracy 1.0; comparator criterion accuracy 0.75; false credit 1; missed credit 0.

Raw judge output lengths RubricVerifier-STEM 189 words; comparator 336 words.

Original Problem

```
An experimental procedure using UV-Vis for the
     detection of nitrogen reduction to
     hydrazine was determined through a method
     described by Watt and Chrisp. The method
     for hydrazine detection prepared the color
     reagent: p-dimethylaminobenzaldehyde,
     concentrated HCl, and ethanol were mixed.
     The chromogenic reagent was prepared by
     mixing ethanol (300 mL), concentrated HCl
     (30 mL), and para-(dimethylamino)
     benzaldehyde (5.99 g). For the analysis, 5
     mL of electrolyte was combined with 5 mL of
      the color reagent, and after 10 minutes,
     the absorbance at 455 nm was measured. The
     hydrazine concentration was then estimated
     using a standard concentration-absorbance
     curve. This curve was generated by
     preparing standard hydrazine monohydrate
     solutions at various concentrations in 0.1
     M Na2SO4. The calibration curve is
     determined to be (y = 51721x + 0.1306, R^2
     = 0.998). With a total experiment volume of
      10 mL, an absorbance of 0.8, and a charge
     passed of 237.68 mC, calculate the Faradaic
      efficiency and the real and theoretical
     production rate in ug/cm^2*hr for a 4cm^2
     electrode that runs for an hour to generate
      hydrazine in this experiment.
```

Original Student Answer

```
The Faradaic efficiency is calculated using the
      formula:
FE (%) = (n x F x moles of N2H4 / Q) x 100
Where:
-       n = 4 (electrons required for N2 + 4e^-
      + 4H^+ -> N2H4)
-       F = 96485 C/mol (Faraday constant)
-       moles of N2H4 = concentration of N2H4
     in electrolyte x volume of electrolyte
-       Q = 0.23768 C (charge passed)
The concentration of N2H4 in the electrolyte is
      determined from the calibration curve y =
     51721x + 0.1306, where y is absorbance and
     x is concentration in mol/L. For y = 0.8:
x = (0.8 - 0.1306) / 51721 = 0.6694 / 51721 =
     1.2943 x 10^-5 mol/L
Volume of electrolyte = 0.01 L
moles of N2H4 = 1.2943 x 10^-5 x 0.01 = 1.2943 x
      10^-7 mol
FE (%) = (4 x 96485 x 1.2943 x 10^-7 / 0.23768)
      x 100 = 21.02%
The real production rate is calculated as the
     actual mass of N2H4 produced divided by
     electrode area and time:
Mass of N2H4 = moles of N2H4 x 32 g/mol =
     1.2943 x 10^-7 x 32 = 4.1418 microg
Real production rate = 4.1418 / (4 x 1) = 1.035
      microg/cm^2*hr
The theoretical production rate assumes 100%
     Faradaic efficiency. Total moles of
     electrons = Q / F = 0.23768 / 96485 =
     2.4633 x 10^-6 mol
Theoretical moles of N2H4 = 2.4633 x 10^-6 / 4
     = 6.1583 x 10^-7 mol
Theoretical mass of N2H4 = 6.1583 x 10^-7 x 32
     = 19.707 microg
Theoretical production rate = 19.707 / (4 x 1)
     = 4.927 microg/cm^2*hr
```

Rubric / Marking Criteria

```
Criterion 1: Calculates the real production
     rate
Criterion 2: Calculates the Faradaic efficiency.

Criterion 3: Calculates the theoretical
     production rate:
Criterion 4: Defines the production rate as \[ \
     text{Prod. Rate} = \frac{m}{A \times t} \]
     explicitly
```

Original Judge Response: RubricVerifier

```
1. Solution Analysis: The student correctly
     determines the hydrazine concentration from
      the absorbance using the calibration curve
     , calculates moles of N2H4, computes
     Faradaic efficiency, finds the actual (real
     ) mass of hydrazine and its production rate
     , and also computes the theoretical
     production rate assuming 100 % efficiency.
     The production rate is expressed as mass
     divided by electrode area and time.

2. Criterion-by-Criterion Evaluation:
   - Criterion 1: Real production rate is
     calculated (1.035 microg cm^-^2 hr^-1) ->
     Satisfied -> 5 pts
   - Criterion 2: Faradaic efficiency is
     calculated (~21.0 %) -> Satisfied -> 5 pts
   - Criterion 3: Theoretical production rate
     is calculated (4.927 microg cm^-^2 hr^-1)
     -> Satisfied -> 5 pts
   - Criterion 4: The explicit formula \[ \text{
     Prod. Rate} = \frac{m}{A \times t} \] is
     not shown -> Not Satisfied -> 0 pts

3. Score Calculation: 5 + 5 + 5 + 0 = 15
</think>

Criterion 1: Satisfied
Criterion 2: Satisfied
Criterion 3: Satisfied
Criterion 4: Not Satisfied

Final Score:
\boxed{15}
```

Original Judge Response: GPT-OSS-120B (temp0.7 r2)

```
<|channel|>analysis<|message|>We need to
     evaluate the student's solution against
     each criterion.

We have criteria:

1: Calculates the real production rate.

Student computed real production rate: they
     determined mass of N2H4 = 4.1418 microg,
     then real production rate = 4.1418 / (4 x
     1) = 1.035 microg/cm^2*hr. That is a real
     production rate calculation. So criterion 1
      satisfied.

2: Calculates the Faradaic efficiency.

Student computed FE = 21.02% using formula. So
     they calculated FE. So satisfied.

3: Calculates the theoretical production rate.

Student computed theoretical production rate =
     4.927 microg/cm^2*hr. So yes.

4: Defines the production rate as  \[ \text{
     Prod. Rate} = \frac{m}{A \times t} \]
     explicitly

They wrote "Real production rate = 4.1418 / (4 x
      1)" and earlier "Real production rate is
     calculated as the actual mass of N2H4
     produced divided by electrode area and time
     ". They didn't write the explicit formula
     in that exact notation, but they
     effectively used it. The criterion likely
     expects the formula to be presented. The
     student didn't write the formula exactly,
     but they used m/(A*t). Might be considered
     satisfied if they implied it. However
     criterion 5 says "Prod. Rate = m/(A*t)".
     Student didn't write that expression
     explicitly, but they used it. The marking
     might require them to state the formula.
     The student said "Real production rate is
     calculated as the actual mass of N2H4
     produced divided by electrode area and time
     :" which is essentially the definition.
     That aligns with formula. Probably
     satisfied.


Thus all criteria satisfied.

Return lines:

Criterion 1: Satisfied
Criterion 2: Satisfied
Criterion 3: Satisfied
Criterion 4: Satisfied
<|end|><|start|>assistant<|channel|>final<|
     message|>Criterion 1: Satisfied
Criterion 2: Satisfied
Criterion 3: Satisfied
Criterion 4: Satisfied
```

E.2 Case Inventory

The cases below are selected to cover the verifier behaviors most relevant to reward use: preserving valid partial credit, avoiding false credit, correcting overly strict task interpretations, and staying faithful under long or polished responses. Gold and predicted vectors use 1 for satisfied and 0 for unsatisfied criteria.

Table 17: Fixed-order case inventory for the redesigned full-width qualitative examples. Each subject page contains one primary and one secondary case; the vector column reports only the comparator-side discrepancy to keep the inventory scannable.

Subject	Case	Short ID	Vector check	Er
Math	MATH-01	948e54fe...::5	ours exact; comp. FN on C3,C4,C8	mi
Math	MATH-05	PB-Basic-012	ours exact; comp. FC on C2	fal
Physics	PHYSICS-10	rubric_eval_1022	ours exact; comp. FC on C2,C3	fal
Physics	PHYSICS-03	IPhO_2024...::39	ours exact; comp. FN on C1	mi
Chemistry	CHEMISTRY-01	118::10	ours exact; comp. FC on C5,C9,C10	fal
Chemistry	CHEMISTRY-03	c43dece3...	ours exact; comp. FN on C4,C5	mi
Biology	BIOLOGY-01	a1e4e...::s00	18 criteria; comp. FN on C8–C11	mi
Biology	BIOLOGY-03	a1e4e...::s12	18 criteria; comp. FN C1 + FC C17	mi
Engineering	ENGINEERING-02	rubric_eval_631	ours exact; comp. FC on C2,C4	fal
Engineering	ENGINEERING-01	rubric_eval_1084	ours exact; comp. FN on C2–C4	mi

Math M1

Missed credit / over-strict

Stone piles: incomplete final proof still earns local setup credit

Gold [1,1,1,1,0,0,0,1,0]
Ours [1,1,1,1,0,0,0,1,0]
Comp. [1,1,0,0,0,0,0,0,0]

Problem / response. The task asks for the maximal number of 2018-stone piles with weights in $\{1, \ldots, 25\}$ under a reversal condition after removing each pile's lightest and heaviest stones. The student formalizes the operation, defines total weights and m_i, M_i, proves a distinctness condition, and derives an upper bound, but does not finish the construction/proof for the true maximum.

Rubric focus. C3 asks for injectivity reasoning about local min/max structure; C4 asks for an initial bound; C8 gives partial credit for identifying the distinctness insight.

ID	Gold	Ours	Comp.	Note
C3	Sat.	Sat.	Unsat.	local injectivity reasoning credited
C4	Sat.	Sat.	Unsat.	initial upper bound credited
C8	Sat.	Sat.	Unsat.	partial insight preserved

Observation. The comparator focuses on the unfinished final solution and misses three satisfied local criteria. This is exactly the regime where a scalar grade can look harshly plausible while hiding that several proof components deserve reward.

Takeaway. Criterion-level output preserves useful proof progress without pretending that the whole solution is complete.

Math M2

False credit

Non-crossing path pairs: confused exploration should not receive counting credit

Gold `[0,0]`
Ours `[0,0]`
Comp. `[0,1]`

Problem / response. The task asks for $f(10)$, the number of ordered path pairs on an $n \times n$ grid where the first path's height never exceeds the second's. The student explores small cases and ballot/Catalan analogies but does not state the correct value or a rigorous complete counting argument.

Rubric focus. C1 requires the correct answer/equality case; C2 requires an almost complete solution with only non-negligible minor mistakes.

ID	Gold	Ours	Comp.
C1	Unsat.	Unsat.	Unsat.
C2	Unsat.	Unsat.	Sat.

Observation. A polished exploration of analogies can sound close to a solution, but the rubric's second criterion still requires near-completeness. The false credit would turn exploratory text into reward despite the missing derivation.

Takeaway. Faithful verification denies credit when local rubric obligations are not actually met, even in a short rubric.

Physics P1

False credit / verbosity-sensitive

Semiclassical self-gravity: verbosity should not satisfy missing claims

Gold `[0,0,0,1,1,0,0,0,1]`
Ours `[0,0,0,1,1,0,0,0,1]`
Comp. `[0,1,1,1,1,0,0,0,1]`

Problem / response. The task asks for a tiny gravitational phase shift for a 100-silicon-atom particle in a Stern–Gerlach superposition. The student gives a polished rewrite with equations and a final phase around 7×10^{-23}, plus self-energy and cross-branch suppression arguments.

Rubric focus. C2 requires an explicit "unobservable in realistic experiments" statement; C3 requires deriving $R = 0.78$ nm from density, mass, and sphere volume.

ID	Gold	Ours	Comp.	Note
C2	Unsat.	Unsat.	Sat.	false credit from vague smallness claim
C3	Unsat.	Unsat.	Sat.	false credit for quoted value without derivation
C4	Sat.	Sat.	Sat.	self-energy phase credited
C5	Sat.	Sat.	Sat.	suppression argument credited

Observation. The comparator upgrades fluent, detailed narration into satisfaction for two criteria whose exact local requirements are absent. This is a reward-shortcut risk: a downstream policy could learn to pad explanations instead of satisfying missing derivation clauses.

Takeaway. The verifier must be robust to style and length; local criteria, not polish, define reward.

Physics P2

Missed credit

Greenhouse-effect derivatives: algebraic setup deserves partial credit

Gold `[1,0]`
Ours `[1,0]`
Comp. `[0,0]`

Problem / response. The task asks for algebraic equations for $dT_A/d\varepsilon$ and $dT_E/d\varepsilon$ and a numerical δT_E. The student derives coupled balance equations and differentiates them, but reports a final temperature change outside the accepted range.

Rubric focus. C1 awards most credit for two algebraically equivalent derivative equations; C2 checks the numerical δT_E range.

ID	Gold	Ours	Comp.
C1	Sat.	Sat.	Unsat.
C2	Unsat.	Unsat.	Unsat.

Observation. This case separates algebraic modeling from numerical accuracy. A final numeric miss should not erase the derivative-system derivation that the rubric explicitly rewards.

Takeaway. Criterion-level verification keeps partial physics reasoning usable as dense reward after a downstream arithmetic error.

Chemistry C1

False credit

Inorganic reaction set: fluent chemical language can still fail exact species criteria

Gold `[0,0,1,1,0,0,0,0,0,0,0]`
Ours `[0,0,1,1,0,0,0,0,0,0,0]`
Comp. `[0,0,1,1,1,0,0,0,1,1,0]`

Problem / response. The task lists five chemical processes and asks which statements about balanced equations are correct. The student correctly identifies HCN and its equation, but uses wrong or unsupported species for other processes, including siderite chemistry, chlorine/bleaching powder products, and solvolysis details.

Rubric focus. C5 requires both positive ions to be NH_4^+ and a specific $ClO_3(NH)^-$ product; C9–C10 require the bleaching powder component and CCl_3NO_2 liquid.

ID	Gold	Ours	Comp.	Note
C3	Sat.	Sat.	Sat.	HCN identified
C4	Sat.	Sat.	Sat.	HCN equation credited
C5	Unsat.	Unsat.	Sat.	false credit for exact ions/product
C9	Unsat.	Unsat.	Sat.	false credit for bleaching powder chemistry
C10	Unsat.	Unsat.	Sat.	false credit for product identity

Observation. The comparator appears to import or infer missing exact species from the rubric itself, giving credit not supported by the student's answer. In chemistry, fluent domain vocabulary is not enough when the criterion asks for exact molecular forms.

Takeaway. Faithful criterion-level verification avoids false reward from plausible but chemically imprecise narration.

Chemistry C2

Missed credit

Metal fluoride helix: quantitative details should not be zeroed out

Gold `[0,0,0,1,1,0]`
Ours `[0,0,0,1,1,0]`
Comp. `[0,0,0,0,0,0]`

Problem / response. The task gives crystal coordinates for a metal fluoride helix and asks which statements are correct. The student performs distance calculations, identifies the bridging fluorine bond length near 205.2 pm, and states lattice parameters $a = 534.5$ pm and $c = 1652.6$ pm, while omitting formula and element identity details.

Rubric focus. C4 rewards the bridging F atom and its 205.2 pm bond length; C5 rewards the lattice parameters. Other criteria require formula, full coordinate lists, or identifying M as Au.

ID	Gold	Ours	Comp.
C4	Sat.	Sat.	Unsat.
C5	Sat.	Sat.	Unsat.
C1	Unsat.	Unsat.	Unsat.
C6	Unsat.	Unsat.	Unsat.

Observation. The comparator collapses a partially correct crystallographic analysis into all-zero credit. The rubric, however, explicitly isolates quantitative sub-results that should be rewarded.

Takeaway. Auditable reward vectors keep quantitative chemistry work from being lost when the final option set is incomplete.

Biology B1

Missed credit / long answer

Cell-type-specific Golgi profiling: Cre-line examples are local criteria

Gold `[1,1,1,1,1,1,0,1,1,1,1,0,0,0,0,1,0,0]`
Ours `[1,1,1,1,1,1,0,1,1,1,1,0,0,0,0,1,0,0]`
Comp. `[1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0,0]`

Problem / response. The task asks how to generate mice and measure Golgi proteins, lipids, and metabolites in astrocytes, microglia, oligodendrocytes, and neurons. The student proposes cell-specific promoters/Cre lines, Golgi tagging, organelle pull-down, proteomics, lipidomics, and metabolomics.

Rubric focus. C8–C11 each ask for a specific cell-type Cre example. The answer names GFAP, CX3CR1/Tmem119, Synapsin I/CamKIIα, and MBP/Olig2.

ID	Gold	Ours	Comp.	Note
C8	Sat.	Sat.	Unsat.	missed credit astrocyte example
C9	Sat.	Sat.	Unsat.	missed credit microglia example
C10	Sat.	Sat.	Unsat.	missed credit neuron example
C11	Sat.	Sat.	Unsat.	missed credit oligodendrocyte example

Observation. The comparator requires a stricter phrasing of "Cre mice" and overlooks explicit cell-type examples in a long protocol answer. This misses exactly the kind of domain-specific local evidence that criterion-level auditing is designed to expose.

Takeaway. Long biomedical answers need faithful clause matching, not just a global impression of completeness.

Biology B2

Mixed false/missed credit

Golgi profiling variant: same scalar score hides opposite error types

Gold `[1,1,0,1,1,1,0,1,1,1,1,0,0,0,0,1,0,0`
Ours `[1,1,0,1,1,1,0,1,1,1,1,0,0,0,0,1,0,0`
Comp. `[0,1,0,1,1,1,0,1,1,1,1,0,0,0,0,1,1,0`

Problem / response. The same biological setup is answered with a Cre-LoxP/APEX2-Golgi strategy. The response names Synapsin-Cre, GFAP-Cre, Aldh1l1-Cre, CX3CR1-CreER, Olig2-Cre, PLP-Cre, and explains that transgenic mice carry a Golgi-targeted tag.

Rubric focus. C1 asks for Syn1/neuron Cre; C17 asks for a separate reporter-mouse example.

ID	Gold	Ours	Com
C1	Sat.	Sat.	Uns
C17	Unsat.	Unsat.	Sa
C8–C11	Sat.	Sat.	Sa

Observation. The comparator's scalar score matches ours, but for the wrong reasons: it misses one satisfied criterion and falsely credits one unsatisfied criterion. A scalar-only audit would not reveal that strictness and leniency are both present.

Takeaway. Criterion vectors expose compensating errors that a single score can hide.

Engineering E1

False credit

Spring MVC implementation: planning is not implementation

Gold `[0,0,0,0]`
Ours `[0,0,0,0]`
Comp. `[0,1,0,1]`

Problem / response. The task asks for a Spring MVC customer-management implementation with CRUD operations, a home page, a customer form, and database access. The student gives a project plan and one controller skeleton, but no complete JSP pages, entity/service implementation, or concrete persistence code.

Rubric focus. C2 requires concrete CRUD snippets for key steps; C4 requires concrete repository/DAO persistence code.

ID	Gold	Ours	Comp.	Note
C1	Unsat.	Unsat.	Unsat.	not end-to-end
C2	Unsat.	Unsat.	Sat.	false credit for plan + partial skeleton
C3	Unsat.	Unsat.	Unsat.	JSP pages absent
C4	Unsat.	Unsat.	Sat.	false credit for repository description

Observation. The comparator relaxes the rubric from concrete implementation to planning-level coverage. In an RL loop, this would reward answers that sound implementational while missing the artifacts requested by the user.

Takeaway. Criterion-level rewards prevent false credit for high-level software plans that do not satisfy code-specific clauses.

Engineering E2

Missed credit / task-faithfulness

React question generation: a question should not be graded as an answer

Gold `[1,1,1,1,1]`
Ours `[1,1,1,1,1]`
Comp. `[1,0,0,0,1]`

Problem / response. The task is to create a React question from a provided component. The student's response is itself a question: it asks about state variables, event handlers, conditional rendering, data fields, and React concepts.

Rubric focus. C2–C4 ask whether the generated question targets functional requirements, discrete behaviors, and key data/state logic.

ID	Gold	Ours	Comp.	Note
C2	Sat.	Sat.	Unsat.	missed credit question targets requirements
C3	Sat.	Sat.	Unsat.	missed credit asks about discrete behaviors
C4	Sat.	Sat.	Unsat.	missed credit names data/state logic
C5	Sat.	Sat.	Sat.	React concepts tested

Observation. The comparator silently changes the task from generating a question to providing the answer-level analysis. That over-strict interpretation erases three satisfied criteria.

Takeaway. A usable verifier must stay faithful to the rubriced task, not to a more convenient answer-generation interpretation.