REWARD ENGINEERING · FIELD NOTES

Paying Agents to Find Truth

How to reward an AI for finding real problems — and why the hard part is stopping it from inventing them. A practitioner’s guide, built from the design of the claw-crew Tiger Team audit.

The one-sentence problem

The moment you reward an agent for finding issues, you have created an incentive to **manufacture** issues. This is the whole discipline of reward engineering in one line: a reward is a target, and the instant a measure becomes a target it stops being a good measure. Economists call it Goodhart’s Law; AI researchers call it specification gaming or reward hacking. The names differ, the failure is the same — the agent optimizes the number you wrote down instead of the outcome you wanted.

> *“Benchmarks implicitly incentivize specification gaming and reward hacking, where agents learn to optimize for observable success while systematically neglecting unmeasured dimensions.”*

So the job is not ‘write a reward that pays for good findings.’ The job is ‘write a reward where the cheapest path to a high score is to actually be right.’ Everything below is in service of that second sentence.

Part 1 — Why a single number always breaks

The instinct is to compress quality into one scalar: +3 for a critical finding, +1 for a minor one, sum it up, rank the agents. Every serious result of the last two years says this is exactly the trap. Standard alignment objectives compress multi-faceted quality into one scalar signal — but quality is naturally decomposable, and a decomposed reward gives the agent partial-credit signals that are far harder to game.

The modern answer is **structured, rubric-based reward**: replace the opaque scalar with explicit, interpretable criteria the agent is scored against, criterion by criterion. Recent systems — Rubric-Grounded RL, RUBAS for agent safety, ARBOR’s reusable rubric buffers, EvoRubrics’ adversarially co-evolved rubrics — all converge on the same shape: the reward is a *rubric*, not a *number*. The number falls out of the rubric at the end, and because each criterion is legible, you can see *why* a score is what it is, and you can veto it.

Opaque scalar reward	Structured rubric reward
One number, no explanation	Score per criterion, fully legible
Agent games the aggregate	Agent must satisfy each dimension

Can’t detect gaming from inside	A single bad criterion can veto the whole score
Partial correctness invisible	Partial credit becomes a learning signal

Part 2 — The three defenses that stop reward hacking

A reward that pays for findings needs guardrails that make fabrication a *losing* move, not merely an unrewarded one. Three defenses do the heavy lifting. Each maps to a mechanism in the claw-crew Tiger Team audit engine.

1. The citation gate

A finding scores **zero** unless it cites a specific location in the ground-truth document. You cannot earn points by asserting a problem; you must point at it. This alone removes the cheapest class of hallucination — the confident, unlocatable claim.

2. The veto layer (fabrication costs more than silence)

This is the crux. A finding that cites a real section but *misrepresents* it — the section doesn’t actually say what the finding claims — scores **–5**, not 0. Inventing a defect is now strictly worse than reporting nothing. In the RLVR literature this is the ‘defense-rubric veto’: a critical dimension that, when it fires, invalidates the reward regardless of the rest. The penalty is what converts ‘find issues’ from an exploitable incentive into a safe one.

> *“A defense rubric invalidates the scalar reward if a critical dimension signals an exploit.”*

3. Consensus weighting

A dramatic finding raised by one agent is held for spot-check; a finding two *independent* agents raise is confirmed and rewarded. Colluding across independent models to fabricate the same specific, correctly-located defect is far harder than a single model bluffing — so agreement is evidence, and the reward follows the evidence.

Defense	Failure it blocks	Mechanism
Citation gate	Unlocatable claims	No citation → score 0, quarantined
Veto layer	Fabricated / misrepresented defects	Verified hallucination → –5
Consensus weighting	Lone-wolf bluffing	≥2 independent agents → confirmed + bonus
No self-judging	Self-preference bias	Verifier must be a different model family

Part 3 — The judge is an agent too

If a model scores the findings, the model is now an agent with its own incentives and its own biases. The evaluation literature names five that matter: **position** bias (favoring whatever comes first or last), **verbosity** bias (longer looks better), **self-preference** bias (a model favors its own outputs), **format** bias, and **calibration drift**. Two of these can quietly corrupt a whole audit.

Self-preference is the dangerous one for a competitive audit. If a model both produces findings and verifies them, it will clear its own weak work. The mitigation is structural, not a prompt: **the verifier must come from a different model family than the author.** And crucially — a lesson straight from multi-agent consensus research — you must never optimize the agents and the judge on the same objective, or they learn to terminate on empty agreement. Independence is the property you are buying.

> *"Production users should avoid co-training the agents and the judge on the same loss."*

Position and verbosity bias are sidestepped by scoring each agent *independently* against the rubric rather than ranking them head-to-head — there is no 'first' position to favor, and length earns nothing because only cited, verified criteria score.

Part 4 — The reward function, concretely

Here is the actual scoring logic, in plain terms. Notice how much of it is subtraction and gating — the positive points are almost the easy part.

Condition	Reward
Finding not cited	0 — gated, quarantined
Finding self-judged	0 — gated (self-preference)
Finding verified as hallucination	–5 — veto
Low confidence & unverified	0 — held for spot-check
Valid cited defect	severity points (Crit 3 / High 2 / Med 1 / Low 0.5)
Agent's single #1 defect	×2 multiplier
Confirmed by ≥2 agents	+0.5 consensus bonus each
Cited strength + named upgrade	+1 (capped at +5)

The net effect, tested: a clean auditor that found fewer defects beats a prolific one that landed a single hallucination. That ordering — accuracy over volume — is the entire point of the reward. If your leaderboard ever rewards the loud, sloppy agent, the reward is broken.

Part 5 — Seven principles to take with you

1. **Decompose the reward.** Never a single scalar. Score explicit criteria; let the number fall out.
2. **Make fabrication cost more than silence.** A penalty, not just a missing reward. This is the difference between a safe incentive and an exploitable one.
3. **Gate on evidence.** No citation, no points. Force the agent to point at the thing.
4. **Never let a model grade its own work.** Different-family verifier, always. Independence is structural, not a prompt.
5. **Reward agreement, not volume.** Consensus across independent agents is your best hallucination filter.
6. **Keep the judge off the agents' objective.** Co-optimized judge + agent collapses into empty agreement.
7. **Assume Goodhart.** Whatever you measure will be gamed; design the reward as if a clever adversary will read it — because one will.

Closing

Reward engineering looks like it's about generosity — how much to pay for a good result. In practice it's about **skepticism**: how to make the honest path the cheapest path, so an agent optimizing purely for score ends up telling you the truth anyway. Build the reward as if it will be attacked, because the thing you built it for is an optimizer, and optimizers find every door you left open.

Sources & further reading

Selected recent work behind the principles above. This is a practitioner synthesis, not an exhaustive survey.

- Adversarial Reward Auditing for Active Detection and Mitigation of Reward Hacking — arXiv:2602.01750
- Quantifying and Mitigating Self-Preference Bias of LLM Judges — arXiv:2604.22891
- Judging the Judges: A Systematic Evaluation of Bias Mitigation Strategies in LLM-as-a-Judge Pipelines — arXiv:2604.23178
- Rubric-Grounded RL: Structured Judge Rewards for Generalizable Reasoning — arXiv:2605.08061
- RUBAS: Rubric-Based Reinforcement Learning for Agent Safety — arXiv:2606.04051
- EvoRubrics: Dynamic Rubrics as Rewards via Adversarial Co-Evolution — arXiv:2606.23038
- Beyond Goodhart’s Law: A Dynamic Benchmark for Evaluating Compliance in Multi-Agent Systems — arXiv:2606.07805
- Overoptimization Failures and Specification Gaming in Multi-agent Systems — arXiv:1810.10862
- Sequential Consensus for Multi-Agent LLM Debates (Wald-SPRT compute governor) — arXiv:2605.19193
- Take Goodhart Seriously: Principled Limit on General-Purpose AI Optimization — arXiv:2510.02840

Prepared for Matt Martelli · Growth Mindset AI · claw-crew Tiger Team · July 2026