

Self-Distilling Reward Models from Asymmetric Information

A General Framework for On-Policy Grounded Self-Reward

Rohan Sikand · Stanford University

March 2026

The Reward Model Problem

Every RL post-training paradigm needs a reward signal. Every current approach has a fundamental flaw:

Paradigm	Source	Flaw
Learned RM (RLHF)	Human preferences	Hackable (Goodhart)
External LLM judge	Prompted frontier model	Distributional gap
Self-reward	Policy judges itself	Reward collapse
Deterministic verifier	Binary pass/fail	Sparse, no partial credit

Learned RMs are proxies that can be exploited. External judges fall behind the policy. Self-reward hallucinates. Verifiers are coarse.

What if you could distill a reward model out of the policy itself — one that's on-policy, dense, and grounded in reality?

The Core Observation

In many training settings, there exists **asymmetric information**: something available *after* the model generates its output that was not available *during* generation.

Setting	Available at generation	Available post-hoc
Code RLVR	Problem statement	Execution feedback
Math RLVR	Problem statement	Interpreter output
RLHF	User prompt	Human preference + rationale
LLM-as-judge	User prompt	Judge critique
RAG / QA	Query	Retrieved documents
Agent training	Task description	Tool API responses
Formal verification	Theorem statement	Proof checker messages

In every case, the post-hoc information resolves ambiguity about output quality. **The same model**, conditioned on this information, becomes a better evaluator.

The General Framework

Principle: Any time asymmetric post-hoc information I exists, you can distill a reward model from the policy itself.

Same model π_θ , two roles, differing only in input:

$$\underbrace{\pi_\theta(\cdot \mid s)}_{\text{Student: generates output}} \qquad \underbrace{\pi_\theta(\cdot \mid s, a_i, I_i, p_{\text{judge}})}_{\text{Teacher: evaluates output given } I}$$

The teacher sees the output a_i plus the asymmetric information I_i . It produces a dense reward:

$$R_{\text{self}}(s, a_i) = \sum_{k \in \mathcal{K}} w_k \cdot P_\theta(q = k \mid s, a_i, I_i, p_{\text{judge}})$$

Why Asymmetric Information is the Key Concept

Without asymmetric information: the teacher sees exactly what the student saw. The teacher can only evaluate based on “intuition” — same knowledge, same blind spots. This is ungrounded self-reward, which collapses.

With asymmetric information: the teacher sees something the student couldn't. This creates an *information advantage* that is:

- **Objective:** the information comes from outside the model (environment, human, tool)
- **Diagnostic:** it reveals *why* the output is good or bad, not just that it is
- **Unforgeable:** the model cannot hallucinate stack traces or human preferences

The quality of the self-distilled reward model scales with the **informativeness of I** :

Label-Free Maintenance (General Form)

The teacher must stay calibrated as θ evolves during training. Two self-supervised objectives, general across all instantiations:

Information prediction — predict I_i from (s, a_i) alone:

$$\mathcal{L}_{\text{pred}}(\theta) = -\mathbb{E} \left[\frac{1}{|I_i|} \sum_k \log \pi_{\theta}(I_{i,k} \mid s, a_i, I_{i,<k}) \right]$$

Forces the model to internalize the causal relationship between outputs and post-hoc information. *Generalizes across all I* : predicting execution feedback, predicting human reactions, predicting tool responses, predicting checker messages.

Calibration — anchor R_{self} to whatever coarse reward exists:

Why Reward Shaping, Not Distillation

A natural alternative: the teacher generates a corrected output conditioned on I , student matches it via KL.

Three reasons reward shaping is the right transfer channel:

1. Asymmetric information creates evaluative advantage, not generative advantage.

Seeing “IndexError at line 42” helps you *judge* code, not *write* code. Seeing a human preference rationale helps you *evaluate* responses, not *generate* them. The teacher is a better judge, not a better generator. Reward shaping exploits evaluative advantage.

2. Distillation collapses diversity.

Distributional matching is mode-seeking: student converges to teacher’s specific output. Reward shaping defines a landscape the student explores freely. Multiple diverse outputs

Instantiation 1: RLVR with Execution Feedback (Primary)

Asymmetric information: Stack traces, compiler errors, partial test outputs

Why RLVR is the cleanest instantiation:

- Feedback is richest: structured, deterministic, interpretable
- Bottleneck is most acute: binary reward with zero-gradient dead zones
- Verifier provides perfect R_{coarse} for calibration
- Two feedback modalities for generality: stack traces (code) + interpreter errors (math)

Judgment classes (domain-specific):

Class	Weight	Grounded in
correct	+1.0	All tests pass

Instantiation Map: Bring Your Own Asymmetric Information

Setting	Asymmetric I	$\mathcal{L}_{\text{pred}}$ becomes	R_{coarse}	Status
Code RLVR	Execution feedback	Predict stack traces	Binary pass/fail	<i>This paper</i>
Math RLVR	Interpreter output	Predict errors	Binary correct	<i>This paper</i>
RLHF	Preference rationale	Predict human reaction	Preference score	Future
LLM-as-judge	Judge critique	Predict judge response	Judge score	Future
RAG / QA	Retrieved docs	Predict relevant passages	Factuality check	Future
Agents	Tool responses	Predict API output	Task success	Future
Formal verif.	Checker messages	Predict proof status	Valid/invalid	Future

Each row is a complete instantiation of the framework. The mechanism is identical: teacher sees I , produces dense reward, $\mathcal{L}_{\text{pred}}$ predicts I , $\mathcal{L}_{\text{cal}}$ anchors to R_{coarse} .

The practitioner's question is just: **what post-hoc information do you have?**

Theoretical Properties (General)

These hold for **any** instantiation, not just RLVR:

Proposition 1: Gradient variance reduction

Dense reward from any grounded source eliminates gradient dead zones in group-relative optimization. $\text{Var}[\hat{A}_{\text{dense}}] \geq \sigma_R^2/G > 0$ regardless of the coarse reward's pass rate.

Proposition 2: Information-theoretic bound

$I(Q^*; I \mid R_{\text{coarse}}, a) > 0$ for any informative asymmetric information. This upper-bounds the improvement from leveraging I . Richer I = higher bound = more room for improvement.

Proposition 3: Calibration safety

If $\mathcal{L}_{\text{cal}} = 0$, optimal policies under $R_{\text{self}} \subset$ optimal policies under R_{coarse} . Dense reward

Multiple Benefits from One Mechanism

The framework yields several downstream benefits simultaneously, in any instantiation:

Dense reward

- Partial credit for near-correct outputs
- Non-zero gradient in all-same regimes
- Failure-mode-specific shaping

On-policy adaptation

- Zero distributional gap
- Teacher evolves with student
- No evasion possible

Reward hack suppression

- Teacher reads the evidence in I
- Can detect exploitative outputs
- $\mathcal{L}_{\text{pred}}$ constrains internal model

Compute efficiency

- One forward pass per sample
- No external model or API
- Smaller group sizes sufficient

These are not separate contributions — they all follow from: “the policy evaluates itself, grounded in asymmetric information.”

What Makes This a Framework, Not a Technique

A technique	A framework
Solves one specific problem	Defines a pattern applicable across settings
“Use execution feedback for dense reward”	“Use any asymmetric information for self-reward”
Validated on one domain	Instantiation map across 7+ domains
Ad hoc design choices	Principled: $\mathcal{L}_{\text{pred}}$, $\mathcal{L}_{\text{cal}}$, judgment classes
One set of results	Theory that holds for all instantiations
Incremental	Opens a research direction

The paper validates the RLVR instantiation. The framework invites the community to instantiate it for their own settings. The question shifts from how do we solve dense reward for RLVR?'' to what asymmetric information is available in your training loop, and have you tried using it?"

Experimental Validation (RLVR Instantiation)

Two feedback modalities demonstrating cross-domain applicability:

- Code: LeetCode / LiveCodeBench — stack traces, compiler errors
- Math: MATH / AIME — interpreter errors, assertion failures

Six baselines spanning all reward paradigms:

Method	Dense?	Adapts?	Paradigm
GRPO (binary)	No	N/A	Verifier
Test-case fraction	Yes	N/A	Verifier (dense)
External judge (Haiku)	Yes	Static	External judge
Ungrounded self-reward	Yes	On-policy	Self-reward
EGSR + labeled buffer	Yes	On-policy	Ours (w/ labels)
EGSR (label-free)	Yes	On-policy	Ours

Key Experiments

E1: Signal validation — Does asymmetric information improve self-judgment?

Teacher AUC with vs. without I (execution feedback), in both domains. *Validates the framework's premise.*

E2: Dense reward effectiveness — Does the self-distilled RM improve training?

Pass@1, convergence speed, within-bucket analysis. *Validates the RLVR instantiation.*

E3: Judge maintenance — Does $\mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{cal}}$ maintain the self-RM without labels?

Teacher AUC over training steps. Label-free vs. labeled buffer. *Validates the maintenance mechanism.*

E4: Reward hack detection — Can the grounded self-judge detect exploits?

Hack rate over training. *Validates the secondary application.*

The RLHF Instantiation (Future Work, Briefly)

To show the framework isn't limited to RLVR:

Setting: RLHF with pairwise human preferences.

Asymmetric I : The human's preference choice + written rationale ("Response A is better because it's more specific about the tax implications").

Teacher: $\pi_{\theta}(\cdot \mid s, a_i, I_{\text{preference}}, p_{\text{judge}})$ — the model reads the human rationale and evaluates other responses.

$\mathcal{L}_{\text{pred}}$: Predict human preference rationale from (s, a_i) alone. Forces the model to internalize what humans care about.

$\mathcal{L}_{\text{cal}}$: Anchor R_{self} to the scalar preference score.

What this gives you: An on-policy reward model that densifies sparse human

The Agent Instantiation (Future Work, Briefly)

Setting: Training an LLM agent that uses tools (APIs, databases, code execution).

Asymmetric I : Tool responses — API return codes, database query results, execution outputs. The agent didn't know what the API would return when it decided to call it.

Teacher: Sees the agent's action plan + all tool responses. Can judge: "This plan failed because the API returned a 404 — the agent queried the wrong endpoint."

$\mathcal{L}_{\text{pred}}$: Predict tool responses from the action plan alone. Forces the agent to build an internal model of tool behavior.

Dense reward: Shaped reward for partial plans — "3 of 5 API calls were correct, the plan structure was sound but the data extraction step failed."

This is where the framework has the most practical value: agent training currently relies on sparse task-completion reward. Tool responses are rich asymmetric information that

Positioning: What We Contribute to the Field

The question shifts from:

“How do we build better reward models?”

To:

*“What asymmetric information is already available
in your training loop, and are you using it?”*

The framework says: you probably don't need a separate reward model. The policy itself, conditioned on information it couldn't see at generation time, is already a reward model. You just need to (1) identify the asymmetric information, (2) prompt the policy to judge conditioned on it, and (3) maintain the judge via $\mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{cal}}$.

Honest Assessment of This Framing

Pros:

- Highest ceiling: opens a research direction, not just a technique
- Unifies multiple ideas under one principle (asymmetric information)
- Theory is prove-once, apply-everywhere
- Invites community contribution (“bring your own I ”)
- Positions the paper as a peer of RLHF/DPO/self-reward paradigms

Cons:

- Generality claim with only RLVR experiments (code + math)
- Reviewer: “you claim generality but only test one setting”
- Risk of being too abstract — concrete results matter more than frameworks

Recommended Paper Structure

1. **Introduction:** The reward model problem across paradigms. Asymmetric information as the unifying observation.
2. **Framework:** General mechanism (1 page). Bring your own I . $\mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{cal}}$. Instantiation map.
3. **RLVR Instantiation:** EGSR specifics. Judgment classes. GRPO integration. Algorithm.
4. **Theory:** Three propositions (framework-level, not RLVR-specific).
5. **Experiments:** E1–E5 on code + math. Concrete, thorough, honest.
6. **Discussion:** RLHF and agent instantiations as future directions. Limitations. “Why not distillation” paragraph.
7. **Conclusion:** The question shift — from building RMs to using available information.

Summary

The framework: Self-distilling reward models from asymmetric information. Any time post-hoc information I exists, the policy itself becomes an on-policy, grounded, dense reward model.

The mechanism: Same model, two roles. Teacher sees I , produces dense reward. $\mathcal{L}_{\text{pred}}$ maintains the internal model of I . $\mathcal{L}_{\text{cal}}$ anchors to coarse ground truth.

The validation: RLVR with execution feedback (code + math). Two feedback modalities, two model scales, comprehensive baselines.

The invitation: Bring your own asymmetric information. The framework is the same. What post-hoc signal are you currently discarding?

Every training loop produces information that's currently thrown away.