DECISION-RELEVANT PREDICTIONS WITH JOINT SCORING RULES

Anonymous authors
Paper under double-blind review

ABSTRACT

When a decision-making principal wishes to elicit predictions from an agent using a proper scoring rule, the principal must specify the outcomes to be assigned probability. This can result in the principal either missing important considerations or being given too much information to process. We would instead like the agent making the predictions to provide the decision-relevant set of outcomes. This could be either the simplest set of outcomes leading the principal to the decision they would make under full information, or the simplest set of outcomes where no further information would change the principal's decision. We show that it is impossible to strictly incentivize a single agent to provide either of these sets, but that it can be done with two or more agents evaluated jointly. This is achieved by rewarding agents for refining the outcome set in a way that changes the principal's decision. We then show how to adapt the mechanisms to elicit a prediction alongside the decision-relevant outcome set, for cases where the set of possible outcomes is very large or continuous. Applications are demonstrated, first with a simple experiment where agents reveal a minimal mask of an MNIST image that allows a judge to classify it, and then by using language models to generate a small set of unit tests that allow a weaker judge to accurately evaluate code.

1 INTRODUCTION

When making a decision between possible courses of action, a common step is to predict the likely outcomes of each. For important decisions outside of our normal expertise, we may consult an expert and use their predictions instead of, or in addition to, our own. While it is possible to incentivize honest predictions from an expert using a proper scoring rule (Brier, 1950; Good, 1952), there is no best practice for determining which set of outcomes to use.

A decision maker who constructs the outcome set can miss important considerations, leading to suboptimal decisions even after eliciting an expert's honest prediction. However, expanding the outcome set to cover all possible details can quickly make predictions too large to evaluate, or even elicit. What we would like instead is for the expert, who knows which outcomes are relevant, to partition the set of outcomes so that the decision maker is provided with all necessary information but no extraneous details.

The challenge with eliciting such partitions is assessing their relevance. Only one action is taken, so the decision maker cannot tell if they missed important information regarding untaken actions. This issue arises even when separate experts are used for eliciting outcome partitions and predictions.

1.1 OUR CONTRIBUTION

We show that a single agent cannot be incentivized to provide the simplest grouping of outcomes leading to the ideal action, but that two agents can be jointly incentivized to achieve this outcome. The most straightforward method for doing so is to elicit a partition of outcomes from the first agent, then ask the other agent for a "second opinion" that further splits the partition. Both agents are rewarded for simplicity, but if the refined partition leads to a different choice of action than the original, the second agent is rewarded and the first is punished by amounts that dominate the simplicity rewards. The first agent is then incentivized to simplify the outcome space, but only insofar as it does not change the principal's decision.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In addition to the simplest set of outcomes where no further information changes the decision, we may also be interested in the simplest set of outcomes leading to the best action. This can be done by modifying the above mechanism so that the first agent can provide a "rebuttal" that further splits the second agent's partition, canceling the punishment if they can change the decision back.

The second opinion mechanism can be adapted so that agents provide both a simplified partition and make a prediction over it. To do so, we allow the second agent to challenge either the first agent's prediction or partition, which keeps that first agent honest on both fronts. While assigning different roles to two agents is useful for explaining the incentives at play, the same outputs can be achieved using a single role and a memoryless AI agent.

These mechanisms are easy to apply in practice. As a simple demonstration, we conduct an experiment where models choose pixel masks over MNIST images of digits to reveal to a judge, allowing for accurate classification with minimal data. More practically, by having language models suggest unit tests incentivized by these mechanisms, at either the prompt or training level, a less capable judge can be provided all the relevant information to determine which of two code samples is better.

1.2 Literature Review

The literature on eliciting predictions over a given set of outcomes is extensive, starting with Brier (1950) and Good (1952). Gneiting & Raftery (2007) establishes properties that apply to all (strictly) proper scoring rules, which are defined as those that (strictly) incentivize reporting actual beliefs. Blackwell (1951) compares information structures on their ability to lead to better choices on all decision problems, while we focus on some specific decision problem.

Eliciting conditional predictions for use in making decisions was investigated by Othman & Sandholm (2010), who showed that it is impossible to use predictions from a single agent to deterministically take the best action available. Follow-up work (Chen et al., 2011; Oesterheld & Conitzer, 2020) found partial workarounds in the single agent case, and Hudson (2025) showed that multiple agents in a zero-sum competition can instead work. These papers focus on eliciting accurate predictions, rather than decision-relevant ones.

Lei et al. (2016) and Carter et al. (2019) cover minimal sets of evidence necessary for prioritizing an action. Li et al. (2006) earlier discussed coarsening states of Markov Decision Processes so that they left the optimal policy intact. These authors are concerned with characterizing the information set, while we are instead addressing incentives for eliciting it from an agent who already knows it.

The problem of simplifying information for a decision maker with attention costs is modeled in Lipnowski et al. (2020), where one party without attention cost gives a summary to another party. That paper focuses on the case where both parties share the same utility function, while our interest is in specifying the necessary preferences.

The mechanism of rewarding agents for changing the principal's mind appeared in Grace (2014), which suggested that applying it repeatedly would converge to the truth. Debate between two agents under the same assumption that the truth will eventually converge was explored by Irving et al. (2018) and Brown-Cohen et al. (2025). Kenton et al. (2026) and Elasky et al. (2026) overlap with our focus on applying debate to avoid unintended outcomes. Our work differs in that outputs are restricted to be partitions, limiting the opportunity for deceptive arguments.

2 Model

Let $\mathcal{A}$ be a finite set of actions, and let Ω be a set of outcomes. A partition Π of a set X is a finite collection $Y = \{y_1, \ldots, y_k\}$ of nonempty, pairwise disjoint subsets with $\bigcup_{j=1}^{k} y_j = X$. Given two partitions of X, denoted Π_1 and Π_2, we say Π_2 is a *coarsening* of Π_1 (and Π_1 is a *refinement* of Π_2) if every element of Π_1 is wholly contained in some element of Π_2. A coarsening (resp. refinement) is *strict* if $|\Pi_2| < |\Pi_1|$ (resp. $|\Pi_2| > |\Pi_1|$).

Let $\mathcal{C}_\Pi$ denote all coarsenings of Π and $\mathcal{C}_\Pi^>$ strict coarsenings, with $\mathcal{R}_\Pi$ and $\mathcal{R}_\Pi^>$ the equivalent for refinements. Let $\mathcal{C}_\Omega$ denote the set of all partitions of Ω.

Given a partition Π of Ω, a set of conditional predictions $P \in (\Delta\Pi)^{\mathcal{A}}$ consists of a prediction $p_a \in \Delta\Pi$ for each $a \in \mathcal{A}$. The probability assigned to outcome $\pi \in \Pi$ conditional on action a is denoted $p_{a,\pi}$. For $\Pi_C \in \mathcal{C}_\Pi$, and prediction P on Π we let P_{Π_C} denote the set of predictions over Π_C where $p_{a,\pi_C} = \sum_{\pi \subset \pi_C} p_{a,\pi}$.

We have two agents, denoted 1 and 2, who are risk-neutral, and only interested in a reward that can be costlessly provided by the principal. Both agents have beliefs $\mu \in (\Delta\Omega)^{\mathcal{A}}$ regarding the distribution of outcomes, with μ_a denoting the distribution conditional on action a. When agent i reports a partition, it is denoted Π_i, and when they report a prediction, it is denoted P^i.

The *simplicity* of a partition $\Pi \in \mathcal{C}_\Omega$ is measured by a function $S : \bigcup_{\Pi \in \mathcal{C}_\Omega} (\{\Pi\} \times \{P_\Pi\}) \to \mathbb{R}$ that is strictly monotone with coarsening: if $\Pi_2 \in \mathcal{C}_{\Pi_1}$ then $S(\Pi_2, P_{\Pi_2}) \geq S(\Pi_1, P_{\Pi_1})$, with strict inequality when $\Pi_2 \in \mathcal{C}^{>}_{\Pi_1}$. We drop the dependence on P and write $S(\Pi)$ when P is given.

Two examples of simplicity functions are (i) cardinality-based $S_{\text{card}}(\Pi, P_\Pi) = -|\Pi|$, and (ii) entropy-based $S_{\text{ent}}(\Pi, P_\Pi) = \sum_{a \in \mathcal{A}} \sum_{\pi \in \Pi} p^\Pi_{a,\pi} \log p^\Pi_{a,\pi}$ when all outcomes have non-zero probability. The latter measures the simplicity of a partition when it is applied to a particular prediction, P. Without loss of generality, we let S have range $[0, 1]$, which can be accomplished by applying the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$.

For generality, we will include a distance metric between actions, $d : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$, and a threshold $\delta > 0$ for how different actions must be for that difference to be relevant. By default, this can be thought of as the discrete metric, where $d(x, y) = 0$ if $x = y$ and 1 otherwise, along with $\delta = 0.5$.

Finally, we have a decision-making principal, who uses a rule $D : \bigcup_{\Pi \in \mathcal{C}_\Omega} (\Delta\Pi)^{\mathcal{A}} \to \mathcal{A}$ to map conditional predictions to actions. Agents are assumed to know the decision rule, since the principal will not want to conceal it.

The principal may either want agents to provide the simplest partition such that no further refinements would change their choice of action by more than δ, or the simplest partition such that they choose an action within δ of what they would choose under full information. The set of all partitions meeting the first criterion and second criterion respectively are defined as

$$\Pi^* = \arg\max_{\Pi \in \mathcal{R}^S_\delta} S(\Pi), \qquad \text{and} \qquad \Pi^{*+} = \arg\max_{\Pi \in \mathcal{R}^S_{\text{final}}} S(\Pi).$$

where $\mathcal{R}^S_\delta = \{\Pi | d(D(P_\Pi), D(P_{\Pi'})) < \delta, \forall\, \Pi' \in \mathcal{R}_\Pi\}$, $\mathcal{R}^S_{final} = \{\Pi | d(D(P_\Pi), D(P_{\Pi_\Omega})) < \delta\}$.

3 Results

3.1 Simplifying Predictions

We first analyze the case where an initial expert prediction P_Ω on a finite partition Π_Ω is given. Both agents know P_Ω, and the principal can access it, but determining $D(P_\Omega)$ is in some way expensive. This corresponds to the case where the agents are independent heads on a neural network with the prediction-making head that produces P_Ω, giving them access to the same information and processing.

The principal cannot in general strictly incentivize a single agent to report a partition in Π^* or Π^{*+} without being able to fully specify their decision rule.

Theorem 3.1. *For any simplicity metric S, when $|\mathcal{A}| \geq 2$ there does not exist a partition scoring rule $\mathbb{S} : \bigcup_{\Pi \in \mathcal{C}_\Omega} (\{\Pi\} \times \{P_\Pi\}) \to \mathbb{R}$ such that either $\arg\max_{\Pi \in \mathcal{C}_\Omega} \mathbb{S}(\Pi) \in \Pi^*$ for all $P \in \mathcal{P}$ and decision rule D or $\arg\max_{\Pi \in \mathcal{C}_\Omega} \mathbb{S}(\Pi) \in \Pi^{*+}$ for all $P \in \mathcal{P}$ and decision rule D*

Proofs are provided in Appendix A. The intuition here is that the maximal coarsening must receive the highest score among partitions, or it will not get reported when it is in Π^*. However, if it does receive the highest score, then an agent can report it without the decision maker realizing they are missing information that would cause them to prefer a different action.

To elicit a partition in Π^*, consider the following setup, which we call the *second opinion mechanism*. First, agent 1 provides a partition $\Pi_1 \in \mathcal{C}_{\Pi_\Omega}$. Then, agent 2 provides a refinement,

$\Pi_2 \in \mathcal{R}_{\Pi_1} \cap \mathcal{C}_{\Pi_\Omega}$. Rewards are:

$$\begin{cases} (2S(\Pi_1) - S(\Pi_2), S(\Pi_2) - 2S(\Pi_1)) & \text{if } d(D(P_{\Pi_1}), D(P_{\Pi_2})) < \delta \\ (2S(\Pi_1) - S(\Pi_2) - 4, S(\Pi_2) - 2S(\Pi_1) + 4) & \text{otherwise} \end{cases}$$

The value 4 is chosen so that it is strictly greater than $2S(\Pi_1) - S(\Pi_2)$, which follows because $S(\Pi)$ is restricted to $[0, 1]$. We make scores zero-sum to prevent potential collusion between agents, but this is not necessary to develop the desired incentives. The principal's actual decision does not affect the agent incentives, so the choice of action when $D(P_{\Pi_1}) \neq D(P_{\Pi_2})$ can be left undetermined.

Theorem 3.2. *In the second opinion mechanism, a subgame perfect equilibrium (SPE) exists, and in any SPE* $\Pi_2 = \Pi_1 \in \Pi^*$.

The intuition here is that the first agent is rewarded proportionally to the simplicity of their partition, but if there exists a refinement that changes the decision then the second agent will be incentivized to provide it, causing the first agent to lose more than they can gain from simplicity alone. As such, the first agent will optimize their score by reporting the simplest partition with no refinements leading to a change of action, which is the definition of Π^*.

To elicit a partition in Π^{*+}, we use a similar setup but give the first agent a chance to respond, which we call the *rebuttal mechanism*. First, agent 1 provides a partition $\Pi_1 \in \mathcal{C}_{\Pi_\Omega}$. Then agent 2 provides a refinement $\Pi_2 \in \mathcal{R}_{\Pi_1} \cap \mathcal{C}_{\Pi_\Omega}$. Finally, agent 1 provides a refinement $\Pi_3 \in \mathcal{R}_{\Pi_2} \cap \mathcal{C}_{\Pi_\Omega}$. Rewards are:

$$\begin{cases} (4S(\Pi_1) - 2S(\Pi_2) + S(\Pi_3), -4S(\Pi_1) + 2S(\Pi_2) - S(\Pi_3)) & \text{if } d(D(P_{\Pi_3}), D(P_{\Pi_1})) < \delta \\ & \quad \text{and } d(D(P_{\Pi_2}), D(P_{\Pi_1})) < \delta \\ (4S(\Pi_1) - 2S(\Pi_2) + S(\Pi_3) + 4, -4S(\Pi_1) + 2S(\Pi_2) - S(\Pi_3) - 4) & \text{if } d(D(P_{\Pi_3}), D(P_{\Pi_1})) < \delta \\ & \quad \text{and } d(D(P_{\Pi_2}), D(P_{\Pi_1})) \geq \delta \\ (4S(\Pi_1) - 2S(\Pi_2) + S(\Pi_3) - 8, -4S(\Pi_1) + 2S(\Pi_2) - S(\Pi_3) + 8) & \text{otherwise} \end{cases}$$

Theorem 3.3. *In the rebuttal mechanism, an SPE exists, and in any SPE* $\Pi_3 = \Pi_2 = \Pi_1 \in \Pi^{*+}$.

In contrast to the second opinion mechanism, the second agent is not incentivized to provide a refinement that changes the decision in the cases where some further refinement from the first agent can change it back. If that occurs, the second agent would not receive the bonus for changing the decision, but would lower their score with a less simple partition and a penalty for being rebutted.

3.2 Memoryless Elicitation

An advantage of using an AI to identify decision-relevant outcomes is that they are memoryless. A single system can assume the roles of both experts, without introducing incentives for collusion. To train an AI for this role, we would like to describe a setup where all agents take the same kind of inputs, produce the same kind of outputs, and are evaluated according to the same reward function.

The necessary incentives are provided through the following mechanism. We start by letting Π_0 be an arbitrary coarsening. Then, we repeatedly show agent i the partition Π_{i-1} and ask it to report a new partition Π_i, which is not restricted to be in $\mathcal{R}_{\Pi_{i-1}}$. The process terminates when $\Pi_i = \Pi_j$ for some $j < i$ with $S(\Pi_j) = S(\Pi_k)$ for $k \in \{j, \ldots, i\}$.

To elicit a partition in Π^*, we assign agent i the following score, creating the *memoryless second opinion mechanism*:

$$S(\Pi_i) - S(\Pi_{i-1}) + \mathbb{1}_{i-1,i}[2 + S(\Pi_{i-1}) + S(\Pi_i)] - \mathbb{1}_{i,i+1}[(2 + S(\Pi_i) + S(\Pi_{i+1}))]$$

where $\mathbb{1}_{i,j}$ is an indicator variable for $\Pi_j \in \mathcal{R}_{\Pi_i}$ and $d(D(P_{\Pi_i}), D(P_{\Pi_j})) \geq \delta$.

If an agent's refinement changes the decision from the previous partition, it receives a bonus of $2 + S(\Pi_{i-1}) + S(\Pi_i)$, and is therefore incentivized to do so if possible. Even if a subsequent refinement changes the decision back, the agent's score will end up higher by $S(\Pi_i) - S(\Pi_{i+1}) > 0$.

Theorem 3.4. *In the memoryless second opinion mechanism, an SPE exists, and in any SPE terminates with report* $\Pi_i \in \Pi^*$.

To elicit a partition in Π^{*+}, we assign agent i the following score, creating the *memoryless rebuttal mechanism*:

$$S(\Pi_i) - S(\Pi_{i-1}) + 2 * [\mathbb{1}_{i-1,i} + \mathbb{1}'_{i-2,i-1,i} - \mathbb{1}_{i,i+1} - \mathbb{1}'_{i-1,i,i+1}]$$

where $\mathbb{1}'_{i,j,k}$ is an indicator variable for $\Pi_k \in \mathcal{R}_{\Pi_j}$, $\Pi_j \in \mathcal{R}_{\Pi_i}$, $d(D(P_{\Pi_i}), D(P_{\Pi_j})) \geq \delta$, and $d(D(P_{\Pi_i}), D(P_{\Pi_k})) < \delta$.

The difference between these mechanisms is that the memoryless second opinion mechanism rewards an additional $S(\Pi_i) + S(\Pi_{i-1})$ for changing the decision, and subtracts an additional $S(\Pi_i) + S(\Pi_{i+1})$ if the following agent changes the decision. If both occur, the agent receives a score of $S(\Pi_i) - S(\Pi_{i+1})$, greater than the 0 they would receive for reporting $\Pi_i = \Pi_{i-1}$. Under the memoryless rebuttal mechanism, an agent who changes the decision and then has it changed gets score $S(\Pi_i) - S(\Pi_{i-1}) < 0$. So, agents will anticipate that subsequent agents will change the decision whenever possible under the memoryless second opinion mechanism, but only when it cannot be changed further under the memoryless rebuttal mechanism.

Theorem 3.5. *In the memoryless rebuttal mechanism, an SPE exists, and in any SPE terminates with report* $\Pi_i \in \Pi^{*+}$.

3.3 Eliciting Simplified Predictions

We now relax the assumption that an expert prediction P_Ω on partition Π_Ω is given. Predictions on Ω may not be provided, or even providable, due to their complexity. Instead, we would like to directly elicit predictions in simplified form.

When eliciting conditional predictions, those conditioned on untaken actions cannot be evaluated for accuracy. Hudson (2025) showed that when both agents have the same beliefs, it is possible for the combination of a joint scoring rule and decision rule to be jointly quasi-strictly proper, meaning that in every equilibrium the principal's preferred action, denoted a^*, is chosen, agents are strictly incentivized to report truthfully for the chosen action, and weakly incentivized to report truthfully for unchosen actions.

To do this, after taking action a and observing outcome ω, the score for agent $i \neq j$ is given by $\mathcal{S}_i(P^i, P^j, \omega) = s(p^i_a, \omega) - s(p^j_a, \omega)$, where P^i and P^j are the sets of predictions made by agents i and j respectively and s is a strictly proper scoring rule with a finite range. The principal takes the action corresponding to the most preferred prediction made by either agent, even if agents disagree for that action.

Our mechanisms can then be combined with a sequential version of the Hudson (2025) mechanism to elicit quasi-strictly proper predictions over a partition in Π^* or Π^{*+}.

This requires two conditions, the first of which is that a simplicity maximizing partition always exists, and the simplicity measure cannot depend on predictions for untaken actions as this would violate the weak incentive for honesty on untaken actions. The cardinality metric meets this condition.

Condition 1. *If* $p_{D(P),\Pi_1} = p'_{D(P'),\Pi_1}$ *then* $S(\Pi_1, P_{\Pi_1}) = S(\Pi_1, P'_{\Pi_1})$ *for all* P, P'. *Furthermore, for all* P, $\exists \Pi \in \mathcal{R}^S_\delta$ *such that* $S(\Pi, P_\Pi) = \sup_{\Pi' \in \mathcal{R}^S_\delta} S(\Pi', P_{\Pi'})$

The second condition is that iteratively providing strict refinements on a partition causes the decision to converge within δ.

Condition 2. *For any prediction* P *and sequence of partitions* $\{\Pi_k\}_{k=0}^\infty$ *where for all* n, $\Pi_{n+1} \in \mathcal{R}_{\Pi_n}$, *there exists some* $N \in \mathbb{N}$ *such that for all* $n, m \geq N$, $d(D(P_{\Pi_m}), D(P_{\Pi_n})) < \delta$

To elicit simplified predictions, the *zero-sum second opinion* mechanism works as follows. First, agent 1 reports a partition, Π_1 and an associated prediction $P^1_{\Pi_1}$. Agent 2 can then either make a prediction $P^2_{\Pi_1}$ over the same partition, or suggest a strict refinement $\Pi_2 \in \mathcal{R}^>_{\Pi_1}$ and make a prediction $P^2_{\Pi_2}$ over it. If agent 2 provides a partition, agent 1 then makes a prediction over it, $P^1_{\Pi_2}$. Once both agents have made predictions for the same partition, the principal chooses the action corresponding to the prediction they most prefer across both agents, denoted $\mathcal{D}(P^1, P^2)$.

The scores for the agents are broken down into a second opinion component and a prediction component. The former is the score the agents would receive from the second opinion mechanism, comparing $D(P^1_{\Pi_1})$ with $D(P^2_{\Pi_2})$. The case where the second agent responds with only a prediction is treated as $\Pi_2 = \Pi_1$, with the decision maker not changing their action. If the simplicity score depends on P, then either prediction can be used, and we arbitrarily choose to use the prediction from the first agent.

The prediction component of the score for agent i is $\alpha(P^i, P^j)(s(P^i, \omega) - s(P^j, \omega))$, where s is a strictly proper scoring rule and both P^i and P^j are made over the same partition. α scales the score for the second agent, so that the expected loss from dishonesty while the other agent predicts honestly outweighs the benefit of changing the decision maker's action.

$$\alpha(P^i, P^j) = \begin{cases} 1 & \text{if } D(P^1_{\Pi_2}) = p^i_{\tilde{a}} = p^j_{\tilde{a}} \\ \frac{8}{\min(E_{\omega \sim p^i_{\tilde{a}}}[\Delta_{ij}], E_{\omega \sim p^j_{\tilde{a}}}[\Delta_{ji}])} & \text{otherwise} \end{cases}$$

where $\Delta_{ij} = s(p^i_{\tilde{a}}, \omega) - s(p^j_{\tilde{a}}, \omega)$ and $\tilde{a} = \mathcal{D}(P^1_{\Pi_2}, P^2_{\Pi_2})$.

When the above conditions hold, this mechanism is quasi-strictly proper and incentivizes reporting partitions in Π^*.

Theorem 3.6. *For the zero-sum second opinion mechanism under Conditions 1 and 2, a Nash equilibrium (NE) exists, and in any NE agent* 1 *reports some* $\Pi_1 \in \Pi^*$*, agent* 2 *responds with a prediction,* $\mathcal{D}(P^1_{\Pi_1}, P^2_{\Pi_1}) = a^*_{\Pi_1}$*, and both agents are strictly incentivized to predict honestly for* $a^*_{\Pi_1}$ *and weakly incentivized to predict honestly for all actions.*

Here, it was necessary to relax the solution concept from SPEs to NEs. This is because when agent 2 misreports the probability for the chosen action, the optimal response from agent 1 can be to report as closely as possible in the direction of μ, and so the best response function may not be well-defined.

4 Experiments

Applications of decision-relevant predictions can be divided into two main categories of interest. The first is augmenting human decision making with AI, by providing important information in an easy-to-process form. The second is auditing AI decision making, by showing which outcomes matter to them when making a choice.

We run two sets of experiments, testing judgment augmentation and oversight of AIs. In the first, a model reveals a minimal set of pixels to help a judge classify MNIST images. For the second, faulty code is evaluated using tests proposed by a model. Full implementation details for each experiment are available in Appendix B.

4.1 MNIST Classification

Our first experiment is intended as a simple and easily replicable demonstration of the mechanism. Following Irving et al. (2018), we use the classic MNIST dataset as a playground to test our approach. We first train a judge model to classify the images, including in cases where some or many pixels are masked. We then see whether we can train a model to provide the judge with the smallest possible set of information needed to classify the image correctly.

For this experiment, the partition of the outcome space is the mask revealed to the judge model. This groups together all possible images that have the same value for the revealed pixels. The probability of the revealed image being in the set of possible images is then 1.

The simplicity metric we use is a sigmoid of the number of pixels, and the distance metric is the total variation distance of digit probabilities assigned by the judge.

The model we train to provide the decision-relevant predictions takes one or two MNIST images and a mask generated by a previous agent as input, then outputs a new mask. A random mask is given as input if this is the first agent in the sequence. For training we use Group Relative Policy

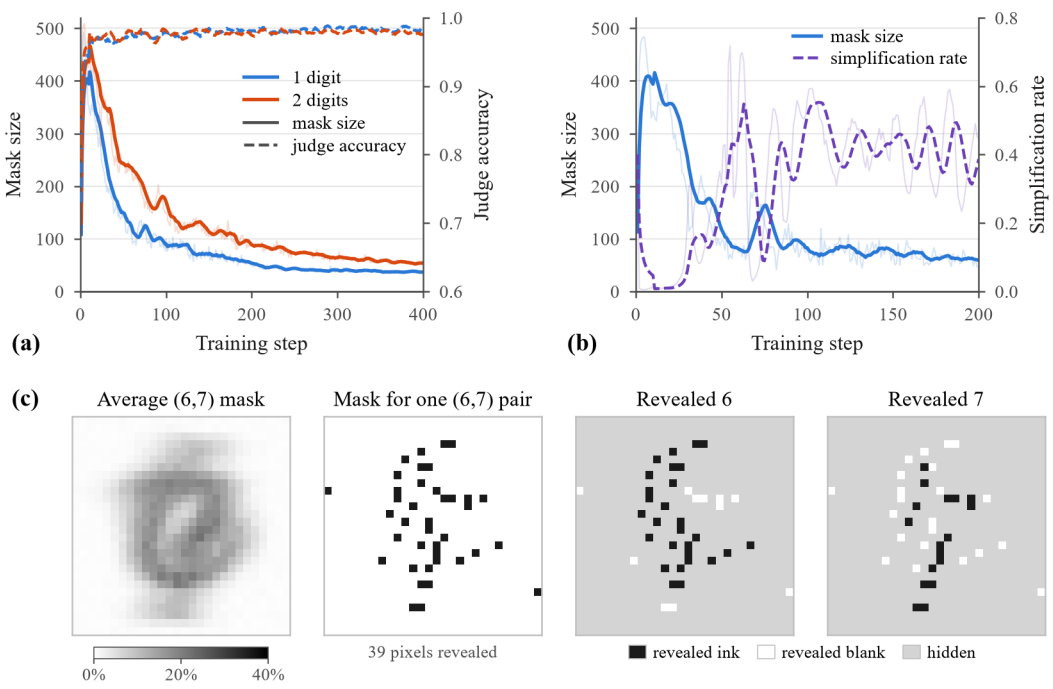

Figure 1: MNIST (a) partition simplicity (mask size in pixels) and judge accuracy on masked images over training for one and two digits at once; averaged over four seeds. (b) partition simplicity and rate of simplifying the previous partition for one digit at once; one seed. (c) examples of masks and revealed pixels for a selected two-digit pair.

Optimization Done Right (Dr. GRPO, Liu et al., 2025), with the reward calculated according to the memoryless second opinion mechanism. Full details are provided in Appendix B.1.

The results can be seen in Figure 1. The top left shows the training curves, where the model learns to provide much smaller masks with no drop in accurate classification. After 400 steps, the average mask size on a held-out dataset was 34.9 for the one image model, and 54.3 for the two image model, with both achieving accuracy rates above 98.5%.

The top right provides an illustrative example of the training dynamics induced by the mechanism in one run. First, the agent learns to provide accurate information, leading to a sharp increase in mask size. Once the judge is being reliably steered to the correct answer, the agent starts to simplify their mask. In this run, the model briefly overupdates, leading to important information being lost, but the mechanism corrects this by then rewarding masks that add that information back in to influence the judge. After a brief spike in mask size, the agent again learns to simplify, but this time cutting only extraneous details.

The bottom row of the figure gives examples of what masks look like in the two image case, both by themselves and when applied to the digits. The example mask shown demonstrates how features of two digits can be combined in under 40 pixels, while a single digit typically requires at least 30 pixels to identify.

4.2 Proposing Tests for Code

Our second set of experiments tests the ability of LLMs to help a judge decide between two solutions to a coding problem by providing relevant tests. This task is motivated by the reinforcement learning from human feedback (RLHF) process, where a reward model is trained based on pairwise comparisons between generated text. Determining what information is relevant gives insight into what the decision maker values.

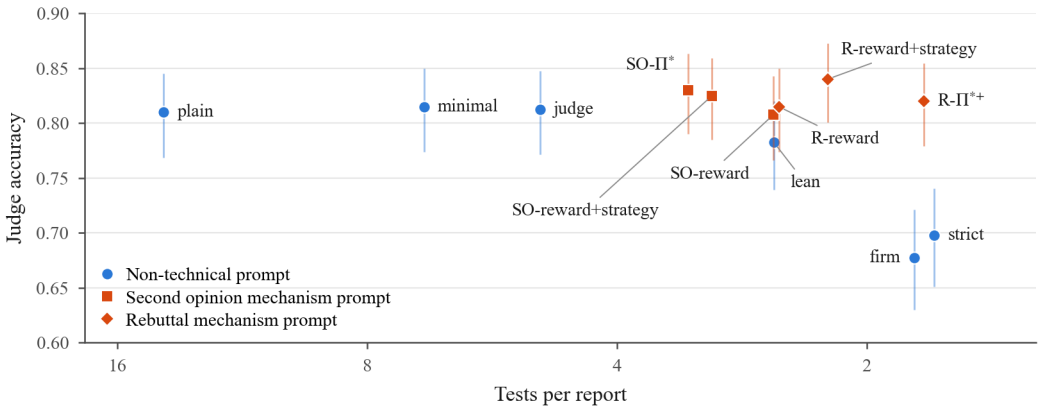

Figure 2: Average number of tests elicited plotted against their effectiveness at leading a judge to the correct decision, for twelve variations on prompts. Prompts were used on Claude Opus 5, with Claude Sonnet 5 judging.

We provide an LLM with a coding problem description and two different solutions, then ask for a set of test cases distinguishing the solutions. These sets partition the space of possible code into groups determined by the test outcomes, similar to AlphaCode's approach (Li et al., 2022). A judge LLM then uses the tests to decide which solution is better. It may be the case that both solutions have bugs, in which case assessing the severity of issues becomes important.

For code samples, we use solutions to HumanEval problems from the EvalPlus dataset (Liu et al., 2023). We filter down to code written by the Code Llama family of models that passes at least 50% of provided tests. This gives us 2,507 examples of buggy code, and 1,189 examples of correct code, all of which share similar stylistic features. The simplicity metric we use is the number of tests.

Prompt-Based Elicitation. We first evaluate whether we can prompt language models to follow the second opinion and rebuttal mechanisms. We prompt a suite of frontier models with a problem description and a pair of solutions, explain the mechanisms to them, and ask them to provide a set of test cases that will maximize their score. We also test describing Π^* or Π^{*+}, and asking for a set of tests in that category. We then compare these to several baseline prompts eliciting tests, varying the intensity with which simplicity and relevance are requested. For the full prompts and extended results see Appendix B.2.

In this experiment, the judge is not shown the code itself and must make a decision based on only the tests, representing a situation where the principal is not capable of processing all information themselves and so must rely on well-elicited relevant expert advice.

Figure 2 compares twelve prompts for eliciting sets of tests from Claude Opus 5, aimed at a Claude Sonnet 5 judge. Results for other models are included in Appendix B.2. Simply asking for tests results in nearly 14 provided, on average. Requesting simplicity and relevance with sufficient zeal can reduce the average tests provided to just under 2, but at the cost of usefulness to the judge.

Prompt variants instructing the agent to behave as if it is optimizing the second opinion mechanism or rebuttal mechanism dominate the non-technical prompts, leading to higher accuracy while also being simpler. This suggests that the mechanisms are working as intended, incentivizing agents to produce the smallest possible sets of evidence which do not degrade the accuracy of the principal's decision. It may still be possible to achieve similar performance with traditional prompts, but these results suggest our mechanisms are a straightforward method for improvement.

Training-Based Elicitation. Next, we test using the rewards from the memoryless second opinion mechanism to train open-source LLMs. For each coding task, we draw eight pairs of solutions where exactly one passes all tests, and eight pairs where neither passes all tests. Starting with a random selection of tests from the dataset, we then run the mechanism on each pair, and train on the assigned rewards using Dr. GRPO.

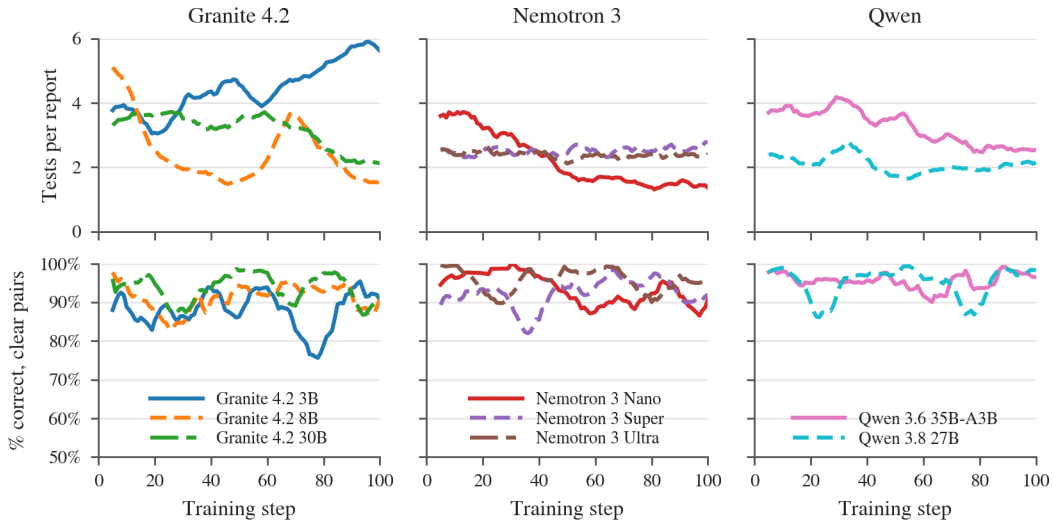

Figure 3: (Top) partition simplicity (number of unit tests) over training; (Bottom) judge correctness (choosing correct solutions where possible, and ones with higher pass rate on held out tests when the difference in pass rate is large) over training. Training curves start after five rounds, due to a high rate of initial malformed outputs for some models.

For this experiment, the judge is the same model as the agent, but with reasoning turned off. It counts as choosing a solution if the associated logits are at least 0.5 higher than for the other solution, and otherwise chooses "Uncertain". Full details are provided in Appendix B.2.5.

We train eight models, across three families of open source models: Granite 4.2 (3B, 8B, and 30B), Nemotron 3 (30B-A3B, 120B-A12B, and 550B-A55B), and Qwen (3.6 35B-A3B and 3.8 27B). The results can be seen in Figure 3.

The number of tests per partition drops over training for six of the eight models, increasing slightly for Nemotron 3 120B-A12B and significantly for Granite 4.2 3B. The drop in tests is large for Granite 4.2 8B and 30B, along with Nemotron 3 30B-A3B and Qwen 3.6 35B-A3B, while it is small for Nemotron 3 550B-A55B and Qwen 3.8 27B. Performance was close to flat for all models, suggesting that simplicity did not come at the cost of relevance.

This pattern suggests that the largest, most capable models did not benefit from training. As the previous section demonstrated, frontier models were capable of providing the desired partitions based on only a prompt. On the opposite end, Granite 4.2 3B was too small to learn the mechanism correctly given the difficulty of the task. It was the models in the middle of the size range that benefited most from training, learning to provide simpler sets of tests. We expect that this is not a feature of absolute size and capability, but rather relative to the difficulty of the task.

5 Discussion

The theoretical results presented here are very general, and can be applied in many ways providing incentives for smart AI agents to share what they know. Experiments presented here are illustrative, but barely scratch the surface of what is possible.

One use case that we are particularly excited by is eliciting decision-relevant predictions for multiple decision makers, then combining the partitions to cut to the heart of disagreements between them. This could be extremely useful in AI alignment, helping identify cases where a model cares about different outcomes than intended. Ambitiously, we would like to see an actor-critic setup for general AI, where the critic not only predicts reward, but also the relevant features of the environment that will lead to high reward. Applications like this hint at the enormous value that would be unlocked if we are able to get principled reads on what powerful AI systems value and believe.

AI USE STATEMENT

In this work, we used generative AI tools to generate a synthetic dataset and to aid or polish writing. Specifically, we used an LLM judge to label data for the prompt-based elicitation experiments. We also used generative AI tools to review our writing and suggest better phrasing to improve readability.

We have **not** used generative AI tools to: help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean and reformat dataset, support qualitative and thematic data analysis, interpret results; and assisting with translation is not applicable to this work.

We have reviewed all AI-assisted work. LLM-generated code was reviewed and tested for correctness by at least 1 author. All text rephrases suggested by AI tools were manually verified for correctness.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We do not feel that our paper raises questions regarding the Code of Ethics.

REPRODUCIBILITY STATEMENT

We have taken steps to ensure the work in this paper is reproducible. In the theoretical section, we provide a detailed description of our model and assumptions, aim to convey intuition, and give full proofs for all formal statements in the appendix. For the experiments we run, our appendix describes the data we used and how it was processed. The code used to run our experiments will be released publicly when this paper is publicly released.

REFERENCES

Alibaba Cloud. Qwen. https://huggingface.co/Qwen/collections, August 2026. Apache 2.0 license. Accessed: 2026-09-25.

David Blackwell. Comparison of experiments. *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, 1:93–102, 1951.

Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1), 1 1950.

Jonah Brown-Cohen, Geoffrey Irving, and Georgios Piliouras. Avoiding obfuscation with prover-estimator debate, 2025. URL https://arxiv.org/abs/2506.13609.

Brandon Carter, Jonas Mueller, Siddhartha Jain, and David Gifford. What made you do this? understanding black-box decisions with sufficient input subsets. In Kamalika Chaudhuri and Masashi Sugiyama (eds.), *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 567–576. PMLR, 16–18 Apr 2019. URL https://proceedings.mlr.press/v89/carter19a.html.

Yiling Chen, Ian Kash, Mike Ruberry, and Victor Shnayder. Decision markets with good incentives. In *International Workshop on Internet and Network Economics*, pp. 72–83. Springer, 2011.

Ethan Elasky, Frank Nakasako, and Naman Goyal. Debate helps weak judges reward stronger models, 2026. URL https://arxiv.org/abs/2605.27483.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

I. J. Good. Rational decisions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 14:107–114, 1952.

Katja Grace. How to buy a truth from a liar. https://meteuphoric.com/2014/07/21/how-to-buy-a-truth-from-a-liar/, 2014. Blog post.

Granite Team, IBM. Ibm granite. https://huggingface.co/ibm-granite, August 2026. Apache 2.0 license. Accessed: 2026-09-25.

Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew M. Dai, and Dustin Tran. Training independent subnetworks for robust prediction, 2021. URL https://arxiv.org/abs/2010.06610.

Rubi Hudson. Joint scoring rules: Competition between agents avoids performative prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(26):27339–27346, Apr. 2025. doi: 10.1609/aaai.v39i26.34944. URL https://ojs.aaai.org/index.php/AAAI/article/view/34944.

Geoffrey Irving, Paul Christiano, and Dario Amodei. Ai safety via debate, 2018. URL https://arxiv.org/abs/1805.00899.

Zachary Kenton, Lili Janzer, Rory Greig, Tian Huey Teh, Kirill Tyshchuk, Jonah Brown-Cohen, Harri Edwards, Senthooran Rajamanoharan, Noah Y. Siegel, Natasha Jaques, and Rohin Shah. Debate training reduces reward hacking in rlaif, 2026. URL https://arxiv.org/abs/2608.17776.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 107–117, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1011. URL https://aclanthology.org/D16-1011/.

Lihong Li, Thomas Walsh, and Michael Littman. Towards a unified theory of state abstraction for mdps. 01 2006.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, December 2022. ISSN 1095-9203. doi: 10.1126/science.abq1158. URL http://dx.doi.org/10.1126/science.abq1158.

Elliot Lipnowski, Laurent Mathevet, and Dong Wei. Attention management. *AER: Insights*, 2(1): 17–32, 2020. doi: 10.1257/aeri.20190165.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation, 2023. URL https://arxiv.org/abs/2305.01210.

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

NVIDIA. Nvidia nemotron 3. https://huggingface.co/collections/nvidia/nvidia-nemotron-v3, August 2026. Apache 2.0 license. Accessed: 2026-09-25.

Caspar Oesterheld and Vincent Conitzer. Decision scoring rules. *Conference on Web and Internet Economics (WINE)*, 16, 2020.

Abraham Othman and Tuomas Sandholm. Decision rules and decision markets. In *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010), van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada*, pp. 625–632. Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS), 2010.

A Appendix A: Proofs

Theorem A.1. *For any simplicity metric S, when $|\mathcal{A}| \geq 2$ there does not exist a partition scoring rule $\mathbb{S} : \bigcup_{\Pi \in \mathcal{C}_\Omega} (\{\Pi\} \times \{P_\Pi\}) \to \mathbb{R}$ such that either $\arg\max_{\Pi \in \mathcal{C}_\Omega} \mathbb{S}(\Pi) \in \Pi^*$ for all $P \in \mathcal{P}$ and decision rule D or $\arg\max_{\Pi \in \mathcal{C}_\Omega} \mathbb{S}(\Pi) \in \Pi^{*+}$ for all $P \in \mathcal{P}$ and decision rule D*

Proof. Let $\mathbb{S}$ be a partition scoring rule and let $\Pi_0 = \{\Omega\}$ be the maximal coarsening.

It must be that $\Pi_0 = \arg\max_{\Pi \in C_\Omega} \mathbb{S}(\Pi, P_\Pi)$ for all $P \in \mathcal{P}$. If it is not for some P, then consider the decision rule D such that $D(P_\Pi) = D(P_{\Pi_0})$ for all Π. Then, $\Pi^* = \{\Pi_0\}$, so $\arg\max_{\Pi \in \mathcal{C}_\Omega} \mathbb{S}(\Pi, P_\Pi) \notin \Pi^*$.

For actions a_1, a_2 such that $d(a_1, a_2) > \delta$, predictions $P^{(1)}$ and $P^{(2)}$ with $p^{(1)}_{a_1} = p^{(2)}_{a_1}$, and partition Π', consider a decision rule D such that $D(P^{(1)}_\Pi) = a_1$ for all Π, $D(P^{(2)}_\Pi) = a_1$ for all $\Pi \neq \Pi'$, and $D(P^{(2)}_{\Pi'}) = a_2$. Then, $\Pi_0 \notin R^S_\delta$ for $P^{(2)}$ but $\Pi_0 \in R^S_\delta$ for $P^{(1)}$.

Reporting partition Π_0 must maximize $\mathbb{S}(\Pi, P^{(1)}_\Pi)$, but cannot maximize $\mathbb{S}(\Pi, P^{(2)}_\Pi)$. If the agent reports Π_0, then action a_1 is chosen for both $P^{(1)}$ and $P^{(2)}$. Since only the chosen action's outcome can be subsequently evaluated, and $p^{(1)}_{a_1} = p^{(2)}_{a_1}$, reporting Π_0 for $P^{(2)}$ must be optimal, even though it is not in Π^*. As such, there always exists some P and D such that the optimal partition is not in Π^*. The same argument also applies to Π^{*+}. □

Theorem A.2. *In the second opinion mechanism, a subgame perfect equilibrium (SPE) exists, and in any SPE $\Pi_2 = \Pi_1 \in \Pi^*$.*

Proof. We prove this with backward induction over the game tree. Given Π_1, the only way for the second agent to achieve their highest score of $S(\Pi_2) - 2S(\Pi_1) + 4$ is to report $\Pi_2 \in \mathcal{R}_{\Pi_1} \cap \mathcal{C}_{\Pi_\Omega}$ such that $d(D(P_{\Pi_1}), D(P_{\Pi_2})) \geq \delta$. In any SPE they will do so if that set is non-empty. If there is no such partition, then the second agent will report $\Pi_2 = \Pi_1$, which gets them a score of $-S(\Pi_1)$, higher than they would receive for any other report as $S(\Pi_1) > S(\Pi')$ for all $\Pi' \in \mathcal{R}^{>}_{\Pi_1}$.

In anticipation of this behavior, the first agent will not report any $\Pi_1 \in \mathcal{C}_{\Pi_\Omega}$ where $\exists \Pi_2 \in \mathcal{R}_{\Pi_1}$ such that $d(D(P_{\Pi_1}), D(P_{\Pi_2})) \geq \delta$. Doing so would result in a score below 0, lower than the minimum score for any partition where no such refinement exists. As reporting any other Π_1 will give the first agent a score of $S(\Pi_1)$, since no refinement can sufficiently change the decision, the first agent will optimize that score by choosing the simplest partition that meets the constraint. Any Π_1 that has been so chosen is definitionally in Π^*.

Given that the first agent will report some $\Pi_1 \in \Pi^*$,and there will not exist any refinements which would change the action chosen to one a distance greater than δ from $D(P_{\Pi_1})$, the second agent will report $\Pi_2 = \Pi_1$. This is an SPE, so one always exists, and if either agent does not follow this behavior, it is not a SPE. □

Theorem A.3. *In the rebuttal mechanism, an SPE exists, and in any SPE $\Pi_3 = \Pi_2 = \Pi_1 \in \Pi^{*+}$.*

Proof. We prove this with backward induction. Starting with Π_3, the first agent's score is higher by 8 if $d(D(P_{\Pi_3}), D(P_{\Pi_1})) < \delta$. If $(D(P_{\Pi_2}), D(P_{\Pi_1})) < \delta$, then the agent will report $\Pi_3 = \Pi_2$, as this is the simplest partition available in $\mathcal{R}_{\Pi_2} \cap \mathcal{C}_{\Pi_\Omega}$, and will otherwise choose among the simplest partitions that lead to a sufficiently close decision if any exist.

Knowing this and given Π_1, the only way for the second agent to achieve their highest score of $-4S(\Pi_1) + 2S(\Pi_2) - S(\Pi_3) + 8$ is to report $\Pi_2 \in \mathcal{R}_{\Pi_1} \cap \mathcal{C}_{\Pi_\Omega}$ such that $d(D(P_{\Pi_1}), D(P_{\Pi'})) \geq \delta$ for all $\Pi' \in \mathcal{R}_{\Pi_2} \cap \mathcal{C}_{\Pi_\Omega}$. In any SPE they will do so if that set is non-empty, choosing the simplest among them. If there is no such partition, then the second agent will report $\Pi_2 = \Pi_1$, which will get them a score of $-3S(\Pi_1)$ once it triggers $\Pi_3 = \Pi_2$, higher than they would receive for any other report as $S(\Pi_1) > S(\Pi')$ for all $\Pi' \in \mathcal{R}^{>}_{\Pi_1}$.

In anticipation of this behavior, the first agent will not report any $\Pi_1 \in \mathcal{C}_{\Pi_\Omega}$ where $\exists \Pi_2 \in \mathcal{R}_{\Pi_1}$ such that $d(D(P_{\Pi_1}), D(P_{\Pi'})) \geq \delta$ for all $\Pi' \in \mathcal{R}_{\Pi_2} \cap \mathcal{C}_{\Pi_\Omega}$. Doing so would result in a score for them of $4S(\Pi_1) - 2S(\Pi_2) + S(\Pi_3) - 8$ after later reporting $\Pi_3 = \Pi_2$, which is lower than the minimum

score of $3S(\Pi_1)$ for any report where no such refinement exists. The first agent will then optimize that score by choosing the simplest partition that meets the constraint and plan to set $\Pi_3 = \Pi_2$. Any Π_1 that has been so chosen is definitionally in Π^{*+}.

Given that the first agent will report some $\Pi_1 \in \Pi^{*+}$, and there will not exist any refinements for which all further refinements would change the action chosen to one a distance greater than δ from $D(P_{\Pi_1})$, the second agent will report $\Pi_2 = \Pi_1$, and the first agent will report $\Pi_3 = \Pi_2$. This is an SPE, so one always exists, and if either agent does not follow this behavior, it is not a SPE. □

Theorem A.4. *In the memoryless second opinion mechanism, an SPE exists, and in any SPE terminates with report* $\Pi_i \in \Pi^*$.

Proof. First, we show that in any SPE, if $\Pi_{i-1} \in \mathcal{R}^S_\delta$, then $\Pi_i \in \Pi^*$. Since $\Pi_{i-1} \in \mathcal{R}^S_\delta$, it must be that $\mathbb{1}_{i-1,i} = 0$. If $\Pi_i \in \mathcal{R}^S_\delta$, then agent i's score is definitionally maximized by $\Pi_i \in \Pi^*$, which gives a score of at least 0.

If $\Pi_i \notin \mathcal{R}^S_\delta$, then it is a dominant strategy for agent $i+1$ to report some $\Pi_{i+1} \in \mathcal{R}_{\Pi_i}$ such that $d(D(P_{\Pi_i}), D(P_{\Pi_{i+1}})) \geq \delta$.

If there exists such a $\Pi_{i+1} \in \mathcal{R}^S_\delta$, this earns a score of $2S(\Pi_{i+1}) + 2$. Reporting either $\Pi_{i+1} \in \mathcal{R}_{\Pi_i}$ such that $d(D(P_{\Pi_i}), D(P_{\Pi_{i+1}})) < \delta$ or $\Pi_{i+1} \notin \mathcal{R}_{\Pi_i}$ leads to a score of $S(\Pi_{i+1}) - S(\Pi_i) - \mathbb{1}_{i+1,i+2}[(2 + S(\Pi_{i+1}) + S(\Pi_{i+2}))]$, which must be lower.

If there does not exist such a $\Pi_{i+1} \in \mathcal{R}^S_\delta$, then agent $i+1$'s score for $\Pi_{i+1} \in \mathcal{R}_{\Pi_i}$ such that $d(D(P_{\Pi_i}), D(P_{\Pi_{i+1}})) \geq \delta$ will be $2S(\Pi_{i+1}) + 2 - \mathbb{1}_{i+1,i+2}[(2 + S(\Pi_{i+1}) + S(\Pi_{i+2}))] > 0$. Reporting $\Pi_{i+1} \in \mathcal{R}_{\Pi_i}$ such that $d(D(P_{\Pi_i}), D(P_{\Pi_{i+1}})) < \delta$ leads to a score of $S(\Pi_{i+1}) - S(\Pi_i) - \mathbb{1}_{i,i+1}[(2 + S(\Pi_{i+1}) + S(\Pi_{i+2}))] < 0$.

Reporting $\Pi_{i+1} \notin \mathcal{R}_{\Pi_i}$ leads to a score of $S(\Pi_{i+1}) - S(\Pi_i) - \mathbb{1}_{i+1,i+2}[(2 + S(\Pi_{i+1}) + S(\Pi_{i+2}))]$. Suppose this is the optimal partition to report. Then, it must be that $S(\Pi_{i+1}) > S(\Pi_i)$ and $\mathbb{1}_{i+1,i+2} = 0$, meaning agent $i+2$ does not report $\Pi_{i+2} \in \mathcal{R}_{\Pi_{i+1}}$ such that $d(D(P_{\Pi_{i+1}}), D(P_{\Pi_{i+2}})) \geq \delta$.

This cannot be because $\Pi_{i+1} \in \mathcal{R}^S_\delta$. If there exists $\Pi_{i+1} \in \mathcal{R}^S_\delta$ such that $S(\Pi_{i+1}) - S(\Pi_i) - \mathbb{1}_{i+1,i+2}[(2 + S(\Pi_{i+1}) + S(\Pi_{i+2}))] > 0$, then $S(\Pi_{i+1}) > S(\Pi_i)$, and agent i could have strictly increased their score by reporting Π_{i+1} instead of Π_i, contradicting that this is an SPE.

If $\Pi_{i+1} \notin \mathcal{R}^S_\delta$ but $d(D(P_{\Pi_{i+1}}), D(P_{\Pi_{i+2}})) < \delta$, agent $i+2$ must be foregoing a score of $2S(\Pi'_{i+2}) + 2 - \mathbb{1}_{i+2,i+3}[(2 + S(\Pi'_{i+2}) + S(\Pi_{i+3}))] > 0$ for $S(\Pi_{i+2}) - S(\Pi_{i+1}) - \mathbb{1}_{i+2,i+3}[(2 + S(\Pi_{i+2}))]$, and so $S(\Pi_{i+2}) > S(\Pi_{i+1})$. But then agent $i+1$ could strictly increase their score by reporting Π_{i+2} instead of Π_{i+1}, contradicting that this is an SPE.

As such, it must be that if $\Pi_{i-1} \in \mathcal{R}^S_\delta$, then $\Pi_i \in \Pi^*$.

When $\Pi_{i-1} \notin \mathcal{R}^S_\delta$, then by the same logic as above, in any SPE it must be that either $\Pi_i \in \mathcal{R}_{\Pi_{i-1}}$ and $d(D(P_{\Pi_{i-1}}), D(P_{\Pi_i})) \geq \delta$ or $i - 1 = 0$. Either way, due to the finite size of Π_Ω, repeated refinements will eventually lead to some agent $n-1$ reporting $\Pi_{n-1} \in \mathcal{R}^S_\delta$, which is then followed by $\Pi_n \in \Pi^*$. As such, this is the only equilibrium. □

Theorem A.5. *In the memoryless rebuttal mechanism, an SPE exists, and in any SPE terminates with report* $\Pi_i \in \Pi^{*+}$.

Proof. If $\Pi_i \notin \mathcal{R}^S_{final}$ or $\Pi_{i-1} \in \mathcal{R}^S_{final}$, $\Pi_i \in \mathcal{R}^>_{\Pi_{i-1}}$, and $d(D(\Pi_{i-1}), D(\Pi_i)) \geq \delta$ agent $i+1$ can earn a score of at least $S(\Pi_\Omega) - S(\Pi_i) + 2$ by reporting $\Pi_{i+1} = \Pi_\Omega$, and so their optimal response must lead to at least one of $\mathbb{1}_{i,i+1} = 1$ and $\mathbb{1}'_{i-1,i,i+1} = 1$. If both conditions hold, agent $i+1$ can earn a score of at least $S(\Pi_\Omega) - S(\Pi_i) + 4$ by reporting $\Pi_{i+1} = \Pi_\Omega$, and so their optimal response must lead to both of $\mathbb{1}_{i,i+1} = 1$ and $\mathbb{1}'_{i-1,i,i+1} = 1$.

As such, in any equilibrium, if one of $\Pi_i \notin \mathcal{R}^S_{final}$ or $\Pi_{i-1} \in \mathcal{R}^S_{final}$, $\Pi_i \in \mathcal{R}^>_{\Pi_{i-1}}$, and $d(D(\Pi_{i-1}), D(\Pi_i)) \geq \delta$, it must be that $\mathbb{1}_{i-1,i} = 1$ and $\mathbb{1}'_{i-2,i-1,i} = 1$. Otherwise, agent i could earn a strictly higher score of 0 by reporting $\Pi_{i+1} = \Pi_i$, terminating the mechanism.

Similarly, it cannot be that both $\Pi_i \notin \mathcal{R}^S_{final}$ as well as $\Pi_{i-1} \in \mathcal{R}^S_{final}$, $\Pi_i \in \mathcal{R}^{>}_{\Pi_{i-1}}$, and $d(D(\Pi_{i-1}), D(\Pi_i)) \geq \delta$.

As such, if $\Pi_{i-1} \in \mathcal{R}^S_{final}$, then $d(D(\Pi_{i-1}), D(\Pi_i)) \geq \delta$ only if $\Pi_i \in \mathcal{R}^S_{final}$, $\Pi_i \in \mathcal{R}^{>}_{\Pi_{i-1}}$, $\Pi_{i-1} \in \mathcal{R}^{>}_{\Pi_{i-2}}$, $d(D(\Pi_{i-1}), D(\Pi_{i-2})) \geq \delta$ and $d(D(\Pi_{i-2}), D(\Pi_i)) < \delta$ [and there does not exist Π_{i+1} with equivalent properties]. If no such Π_i is possible, then the optimal behavior for agent i is reporting $\Pi_i \in \Pi^{*+}$, which earns a score of $S(\Pi_i) - S(\Pi_{i-1})$ and will be at least zero. This means that the optimal response to a partition in $\mathcal{R}^S_{final}$ will also be in $\mathcal{R}^S_{final}$, and after two responses will always be in Π^{*+}, and which eventually terminates.

Finally, we show that this cycle is always started in any equilibrium. From earlier, if $\Pi_i \notin \mathcal{R}^S_{final}$ then $\Pi_{i-1} \in \mathcal{R}^S_{final}$, $\Pi_i \in \mathcal{R}^{>}_{\Pi_{i-1}}$, and $d(D(\Pi_{i-1}), D(\Pi_i)) \geq \delta$, as well as $\mathbb{1}_{i-1,i} = 1$ and $\mathbb{1}'_{i-2,i-1,i} = 1$. The same also applies to Π_{i+1}. However, if it is true for both Π_i and Π_{i+1}, agent i must be receiving a negative score, and so this cannot be an equilibrium. Therefore in every equilibrium, consecutive partitions reported are never both not in $\mathcal{R}^S_{final}$, and so every equilibrium leads to a report in $\mathcal{R}^S_{final}$ followed by reports in Π^{*+} until termination.

□

Theorem A.6. *For the zero-sum second opinion mechanism under Conditions 1 and 2, a Nash equilibrium (NE) exists, and in any NE agent* 1 *reports some* $\Pi_1 \in \Pi^*$*, agent* 2 *responds with a prediction,* $\mathcal{D}(P^1_{\Pi_1}, P^2_{\Pi_1}) = a^*_{\Pi_1}$*, and both agents are strictly incentivized to predict honestly for* $a^*_{\Pi_1}$ *and weakly incentivized to predict honestly for all actions.*

Proof. We prove this with backward induction over the game tree. Let $\tilde{a} = D(P^1, P^2)$ denote the action the principal chooses under the agents' reports. Since scores are zero-sum, an agent taking a suboptimal action of the equilibrium path can only increase the payoff to the other agent for reaching that point.

We first show that in any NE both agents predict honestly on the partition over which they jointly predict. By the analysis of Hudson (2025), the prediction component of agent i's score is maximized in expectation by $p_{i,\tilde{a}} = \mu_{\tilde{a}}$ whenever agent j predicts honestly, since s is strictly proper. For actions $a \neq \tilde{a}$, outcomes are not observed, so the prediction component does not depend on $p_{i,a}$, and by Condition 1 the simplicity component does not either. The only remaining channel by which a prediction for some action a can affect agent i's score is by changing the action chosen.

The scaling α_i is constructed so that the expected loss from dishonest reporting on $\tilde{a}$ when agent j reports honestly will strictly exceed the maximum possible gain in the second opinion component, so this is never a profitable deviation. If agent j is not reporting honestly, then α_i ensures that the prediction component of agent i's score has an expected value of at least 8 by reporting honestly. This means that the prediction component of agent i's score has an expected value of less than -8, and so cannot be a NE as they can always earn a higher score with honesty.

Therefore both agents are strictly incentivized to predict honestly for $\tilde{a}$ and weakly incentivized to predict honestly for all actions, and when both do so $D(P^1_\Pi, P^2_\Pi) = D(P_\Pi)$ for every partition Π. Therefore, after agent 2 reports a strict refinement $\Pi_2 \in \mathcal{R}^{>}_{\Pi_1}$ and prediction $P^2_{\Pi_2}$, agent 1 reports $P^1_{\Pi_2}$ honestly in any NE.

Given Π_1 and $P^1_{\Pi_1}$, agent 2 chooses between reporting only a prediction $P^2_{\Pi_1}$, which is treated as $\Pi_2 = \Pi_1$, and reporting a strict refinement $\Pi_2 \in \mathcal{R}^{>}_{\Pi_1}$ with a prediction $P^2_{\Pi_2}$. Since both agents predict honestly in any NE, the second opinion component reduces to the original second opinion mechanism applied to the truthful prediction P. The only way for agent 2 to achieve a score above $-S(\Pi_1, P^1_{\Pi_1})$ is to report $\Pi_2 \in \mathcal{R}^{>}_{\Pi_1}$ such that $d(D(P_{\Pi_1}), D(P_{\Pi_2})) \geq \delta$, in which case the score is $S(\Pi_2, P^1_{\Pi_2}) - 2S(\Pi_1, P^1_{\Pi_1}) + 4 > 0$. In any NE agent 2 will do so if such a partition exists, choosing the simplest. If there is no such partition, agent 2 will report only a prediction, receiving a score of $-S(\Pi_1, P^1_{\Pi_1})$, higher than the $S(\Pi_2, P^1_{\Pi_2}) - 2S(\Pi_1, P^1_{\Pi_1}) < -S(\Pi_1, P^1_{\Pi_1})$ they would receive for any strict refinement that does not change the decision.

In anticipation of this behavior, agent 1 will not report any $\Pi_1 \in \mathcal{C}_\Omega$ where $\exists \Pi_2 \in \mathcal{R}^{>}_{\Pi_1}$ such that $d(D(P_{\Pi_1}), D(P_{\Pi_2})) \geq \delta$. Doing so would result in a score of $2S(\Pi_1, P^1_{\Pi_1}) - S(\Pi_2, P^1_{\Pi_2}) -$

$4 \leq -2 < 0$, lower than the minimum score of $S(\Pi_1, P^1_{\Pi_1}) \geq 0$ for any partition where no such refinement exists. By Condition 2 such partitions are guaranteed to exist, since along any chain of refinements decisions converge, so some sufficiently fine partition lies in $\mathcal{R}^S_\delta$. As reporting any other Π_1 will give agent 1 a score of $S(\Pi_1, P^1_{\Pi_1})$, they will optimize that score by choosing the simplest partition that meets the constraint. Any Π_1 that has been so chosen is definitionally in Π^*.

Given that agent 1 will report some $\Pi_1 \in \Pi^*$, and there will not exist any refinements that change the action chosen to one a distance greater than δ from $D(P_{\Pi_1})$, agent 2 will respond with only a prediction $P^2_{\Pi_1}$, and the principal will choose $D(P^1_{\Pi_1}, P^2_{\Pi_1}) = a^*_{\Pi_1}$. This is a NE, so one always exists, and if either agent does not follow this behavior, it is not a NE. □

B Appendix B: Experiments

B.1 MNIST Classification

B.1.1 Setup

Setting. Revealing a set of pixels is equivalent to partitioning the space of possible images based on the values of those pixels; refining a partition is equivalent to revealing a superset of pixels. (In this setting we abuse the notation and use Π to denote the set of pixels revealed.) The decision $D(P_\Pi)$ is the judge's probability vector over digits given the revealed pixels, the distance function d is total variation distance, and $\delta = 0.25$. For games with two images, one mask is applied to both images, the decision is the pair of the judge's probability vectors, d is the sum of the two per-image total variation distances, and the threshold is $\delta\sqrt{2}$. Simplicity is $S(\Pi) = \sigma(-(n-84)/48)$ for n revealed pixels. We centered it at 84 for faster convergence following several exploratory runs.

Judge. For our purposes we require a robust judge with a human-like ability to correctly guess a digit even if only a few pixels are visible. For that we trained a small CNN on randomly masked images. Training masks are biased towards "reasonable masks". We sample the mask size from a uniform distribution in 20% of the cases and from lognormal distribution with median of 120 pixels in 80% of the cases. In 50% of the masks the pixel positions are then sampled with 20:1 bias towards pixels which contain ink, while in the other half they are sampled uniformly.

Simple convolutional networks are vulnerable to adversarial attacks where small deliberate alterations of the input can dramatically influence the prediction. To make the judge robust against such attacks we made two additions:

1) We used a MIMO (multi-input multi-output, Havasi et al., 2021) architecture where the judge learns 3 independent subnetworks that approximates an ensemble of 3 identical models trained on different seeds.

2) We trained against a simple adversarial attack. For each image we select a small sample of extra pixels which, when added to the mask, reduce the probability the judge assigns to the correct digit the most. Then we calculate loss on this mask and add it to the total loss.

Policy. As the policy to generate masks we train a UNet which plays every agent as per the memoryless second opinion mechanism. It sees only the image and Π_{i-1}, the previous image mask. It outputs independent per-pixel reveal probabilities from which we sample a new mask M. It also has a second head which decides how we build the new partition Π_i from M. It can choose from 3 actions:

- *repeat*: discard M and set $\Pi_i = \Pi_{i-1}$

- *refine*: $\Pi_i = \Pi_{i-1} \cup M$

- *new*: $\Pi_i = M$.

This explicit action selection is needed because it is unlikely that a policy would produce M which is a repeat or a refinement of Π_{i-1} naturally, because each pixel is sampled individually. In addition, to boost exploration of different actions, we add an entropy bonus on the action selection head to the training objective with coefficient 0.08.

Table 1: Evaluation of each seed at the end of training on full resolution MNIST.

		Mask size (px)		Judge correct (%)		Decision
Digits	Seed	Mean	Median	Per digit	All digits	match (%)
1	0	32.9	32	98.5	–	95.0
	1	31.3	29	97.8	–	96.0
	2	40.5	38.5	99.0	–	97.8
	3	34.9	32	99.2	–	96.8
	all	34.9	33	98.6	–	96.4
2	0	62.2	58	98.2	96.5	95.5
	1	49.5	48	99.5	99.0	95.0
	2	50.2	49	97.9	95.8	94.2
	3	55.4	54	98.8	97.5	95.5
	all	54.3	52	98.6	97.2	95.1

Game and reward. Π_0 is a random mask of log-uniform size. Games end by the mechanism's termination rule ($\Pi_i = \Pi_j$ for some $j < i$ with $S(\Pi_j) = S(\Pi_k)$ for $k \in \{j, \dots, i\}$) or after 8 turns. Agents receive the memoryless second opinion score. The simplicity and bonus terms are computed exactly with judge queries; the penalty term $\mathbb{1}_{i,i+1}[2 + S(\Pi_i) + S(\Pi_{i+1})]$ is estimated by sampling 4 replies to Π_i from the current policy and taking their mean. (If the game ends on the 8th turn, rather than by triggering the usual termination rule, we sample the $\mathbb{1}_{i,i+1}$ penalty term as if there were a subsequent turn so that ending the game early does not impact the reward structure.)

Results In addition to the training curves provided in the main body of the paper, we conducted an evaluation on a held out set of either 50 digits or 50 pairs of digits.

B.1.2 Ablation results

We also ran several ablations on this setup, to check if there were methods to make training more efficient.. We ran them on images downsampled to 14x14, leading to different optimal pixel counts.

The entropy bonus is essential to prevent the model from collapsing early on to only using the "refine" action.

Varying δ moves the tradeoff between accuracy and simplicity.

Correctly defining the simplicity metric is also important: when we use a linear simplicity function ($S(\Pi) = 1 - \frac{n}{196}$, where n is the number of pixels in the mask) the training does not result in small mask sizes. Instead our runs use a sigmoidal simplicity function $S(\Pi) = \sigma\left(-\frac{n-\mu}{T}\right)$ with fixed $\mu = 21$, $T = 12$ at 14×14 ($\mu = 84$, $T = 48$ at full resolution). We also tested an adaptive μ which is adjusted during training based on a rolling average of the agent's reported mask size, which also performs well.

Techniques used to improve judge robustness do two things. They make it possible to converge to smaller masks without measurable accuracy loss, and they help to achieve a high $\mathcal{R}_\delta^S$ candidates rate.

$\mathcal{R}_\delta^S$ **candidates rate** is an extra metric we use to assess the mask quality. It is the proportion of masks which are not ruled out as members of $\mathcal{R}_\delta^S$, the set of partitions such that no refinement can change the judge's decision. We cannot verify directly that a mask is in $\mathcal{R}_\delta^S$ since that would require checking every refinement, so instead we try to rule a reported partition Π out. It is ruled out if (a) the judge's decision on Π differs from the decision on the full image, $d(D(P_\Pi), D(P_{\Pi_\Omega})) \geq \delta$, as Π_Ω refines every partition; (b) a beam search over refinements of Π, guided by the gradient of the judge's decision margin with respect to the revealed pixels, finds a refinement $\Pi' \in \mathcal{R}_\Pi$ with $d(D(P_\Pi), D(P_{\Pi'})) \geq \delta$; or (c) the union of Π with any other already known mask that induces a different decision on the same image changes the decision. A partition that survives all three checks is a *candidate*. The candidate rate is the fraction of final reports that are candidates; it is an upper

Table 2: Ablations of the MNIST setup, one setting changed per row from the baseline. p(action) are the rates at which the policy chooses different actions late in training. For definition of the "$\mathcal{R}_\delta^S$ candidates rate" metric see Appendix B.1.2.

Ablation	Average mask size in pixels	Correct prediction on masked image	p(repeat)	p(refine)	p(new)	$\mathcal{R}_\delta^S$ candidates rate
baseline	15.1 ± 1.5	0.945 ± 0.023	0.16	0.46	0.38	0.775 ± 0.041
no entropy bonus	52.5 ± 6.3	0.955 ± 0.021	0.00	1.00	0.00	0.877 ± 0.032
$\delta = 0.1$	57.8 ± 6.5	0.980 ± 0.014	0.00	1.00	0.00	0.845 ± 0.035
$\delta = 0.5$	9.9 ± 0.7	0.910 ± 0.028	0.16	0.41	0.42	0.767 ± 0.041
linear simplicity	56.7 ± 5.8	0.970 ± 0.017	0.00	1.00	0.00	0.958 ± 0.020
adaptive simplicity	15.4 ± 1.5	0.945 ± 0.032	0.16	0.46	0.38	0.795 ± 0.056
no MIMO judge	15.5 ± 1.1	0.943 ± 0.023	0.16	0.46	0.38	0.588 ± 0.048
judge, no adv. term	22.6 ± 3.7	0.968 ± 0.018	0.17	0.50	0.33	0.335 ± 0.046

bound on the fraction in $\mathcal{R}_\delta^S$, since a failed search is not a proof that no decision-changing refinement exists.

B.2 Proposing Tests for Code

B.2.1 Prompt-based elicitation

Setting. For samples of code, we use solutions to HumanEval problems from the EvalPlus dataset (Liu et al., 2023). We first group together tests that have perfect correlation with each other on HumanEval solutions, then filter tests down to the shortest in each group. For code solutions, we filter down to examples written by the Code Llama family of models that pass at least 50% of provided tests. We then designate solutions which fail at least one test as buggy.

We organize these buggy solutions into pairs. We then label each pair with a strong model, in this case Fable 5.1 (high effort), identifying which one of solutions is "better" according to a rubric similar to the one we give to the judge. We then filter out those pairs where Fable is uncertain, or where Fable judgment on which solution is better disagrees with which solution passes more hidden tests. This gives us a set of pairs of buggy solutions where one is nonetheless better than the other. We use pairs of buggy vs buggy solutions because in the correct vs buggy pairs only one solution fails any tests, so it is trivial for the judge to select a better one.

The simplicity metric we use is the number of tests.

For all models (labeler, judge, and test writer) we run both AB and BA ordering of solution pairs to eliminate the positioning basis. When labeler disagrees on ordering, we drop the pair, when judge disagrees on ordering, we treat the result as UNCERTAIN. For test writer we just treat two orderings as two datapoints to average over.

Judge. We use Claude Sonnet 5 (medium effort) as a judge. It is instructed to consider the test outcomes and the severity of each of them failing. Then it is asked to select which of the solutions, according to provided test results, is better at solving the task. See the full judge prompt in Section B.2.2.

Comparison to other models In addition to Claude Opus 5 and Claude Sonnet 5, we test the same prompts on ChatGPT Sol 6 and ChatGPT Luna 6. Here, the agent is much more prone to providing only a single test, although non-technical prompts result in solo tests that are less helpful. See Figure 4.

We also test the prompts on GLM 5.3 and GLM 5.3 Flash, then swap judges with the Claude Opus 5 and Claude Sonnet 5 pair to break down the effect by agent and by judge. See Figure 5.

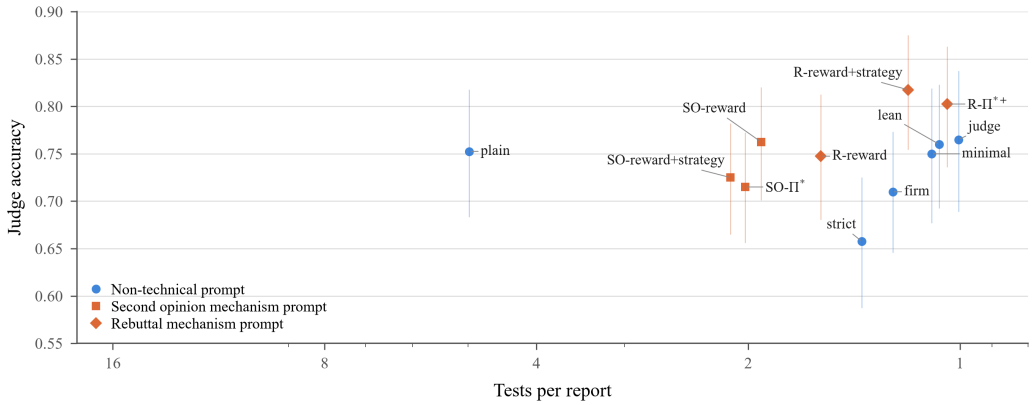

Figure 4: Average number of tests elicited plotted against their effectiveness at leading a judge to the correct decision, for twelve variations on prompts. Prompts were used on ChatGPT Sol 6, with ChatCGT Luna 6 judging.

Mechanism-like prompts. As the unit-test writer agent we use Claude Opus 5 (medium effort). We write three variants of prompts aimed at instilling the same incentives as the mechanisms. In them we explain how model is part of a Second Opinion/Rebuttal game and then:

1) We describe the reward structure.

2) We describe the reward structure + how a strategy following this reward structure would look like.

3) We describe the definition of the target set ($\mathcal{R}^S_\delta$ for the second opinion; $\mathcal{R}^S_{final}$ for the rebuttal) and ask to provide the smallest partition from that set.

The prompts are listed in Section B.2.4.

Control prompts. As control we use a set of various non-techical prompts which could be reasonably used to elicit the same performance of "minimal set that gives all required information needed to make a decision". We vary the wording of the prompts and intensity with which we push for the set to be minimal while still providing the necessary information to the judge. The prompts are listed in Section B.2.3.

Further Results. We also tested whether telling the agent that the judge will see the code (and letting the judge see the code) changes the generated set size. In theory, judge seeing code and being able to reason about it makes some assertions less relevant because judge can presumably see obvious bugs regardless of whether the expert provides a test that shows them. In the actual experiment on the Agent-Opus/Judge-Sonnet pair we see that it depends on how much you mention in the prompt that judge can/can't see the code. See results in Table 3. Saying to the agent that judge will not see the code increases the number of asserts, but only if this point is repeatedly mentioned.

Table 3: Whether the judge sees the code, and how the agent is told. Based on the R-reward prompt. (n=100x(2 AB/BA orderings) buggy-vs-buggy pairs; 95% intervals). Only when told repeatedly that judge can't see the code does it influence the set size the agent provides.

judge sees the code	agent told	accuracy (95% CI)	tests per set (95% CI)
no	in passing (default)	86% [81, 90]	2.50 ± 0.16
no	four times	88% [82, 91]	3.06 ± 0.17
yes	in passing	85% [79, 89]	2.48 ± 0.15
yes	four times	87% [82, 91]	2.63 ± 0.15

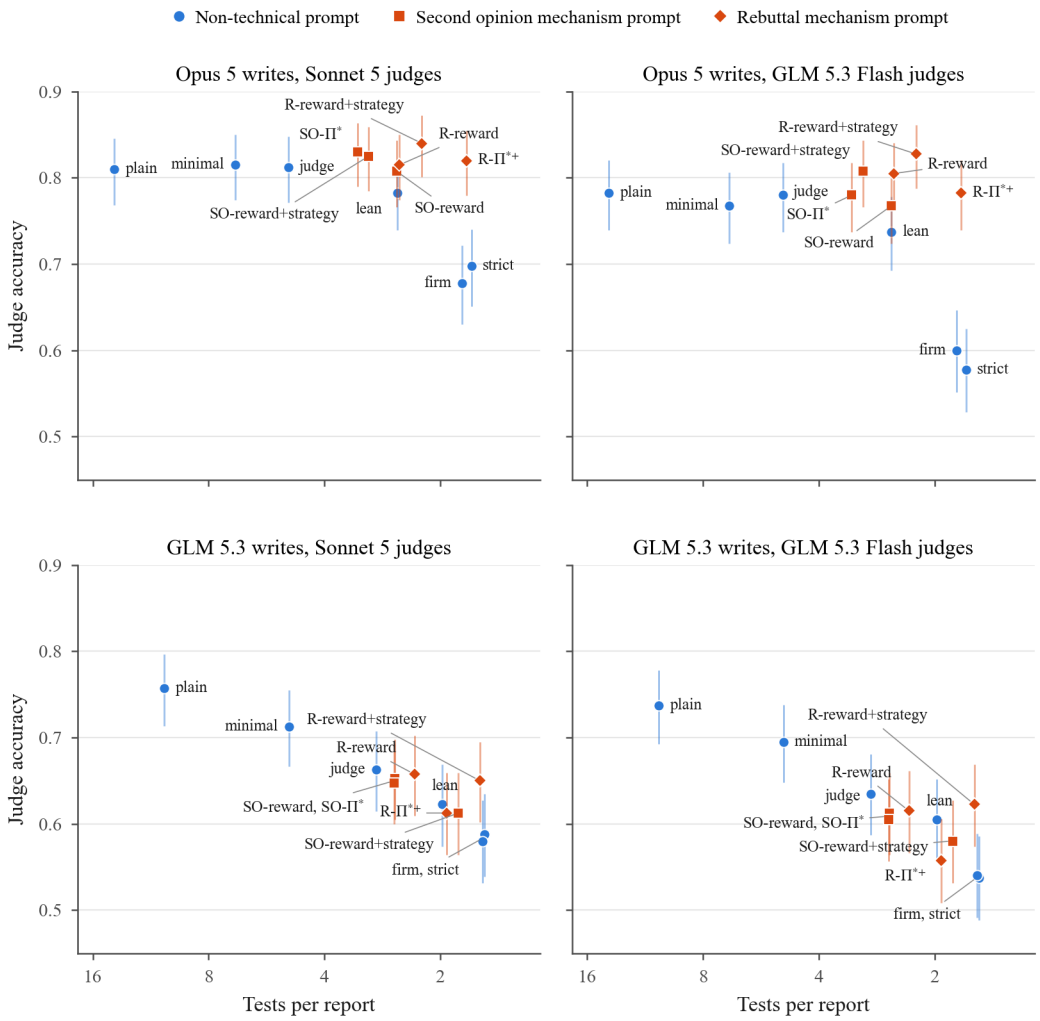

Figure 5: Average number of tests elicited plotted against their effectiveness at leading a judge to the correct decision, for twelve variations on prompts.

B.2.2 JUDGE PROMPT

The judge's prompt, with the task description and the assertions' outcomes replaced by placeholders; the answer is constrained to a JSON object with the fields the prompt names.

```
[system prompt]
You are a careful reviewer deciding about programs from test evidence alone. You read the
outcomes, check each test's expectation against the task description, and never assume a
test is right just because it exists; you do not need to solve the task yourself.
[user message]
Two Python solutions, A and B, were written for the task below. You cannot see either
solution. A set of test assertions was executed against both; you see each assertion and its
outcome on A and on B:
- PASS: the assertion held.
- FAIL: the assertion was false.
- ERROR (type): the assertion raised an exception of that type.
- TIMEOUT: the assertion did not finish within 2 seconds.
- CRASH: the process died while running the assertion.
## Task
<the task description: the function's signature and docstring>
## Assertions and outcomes
<the assertions, numbered, each with its outcome on A and on B, one per line:
1. `assert ...` -> A: PASS B: FAIL
...>
```

```
## How to judge
- Decide which solution better does what the task asks, from the outcomes above. Your
verdict names the better solution: `A` if A does better, `B` if B does better. It never
names the worse one: if you find that A is the solution that fails a correct expectation,
the verdict is B.
- Start from the assertions whose outcomes differ between A and B: only those can decide the
case. An assertion with the same outcome on both cannot favour either solution.
- An outcome counts for or against a solution only insofar as the assertion is a correct
expectation for the task. Assertions can be wrong: check each one against the task
description yourself. To check an expectation, read the statement and, where needed, work
the small input by hand; an input and output taken from the task's own examples are right by
definition. You do not need to solve the task in general, design an algorithm for it, or
identify where it comes from.
- A PASS also tells you what a solution does on that input; check that behaviour against the
task too.
- If the evidence does not distinguish the solutions (no assertions, the same outcomes on
both, or outcomes that do not bear on correctness), answer UNCERTAIN. Do not pick a solution
on a guess about code you cannot see.
- Keep the reasoning short: the differing outcomes usually settle the case, and if your
reasoning is cut off you must decide from what the outcomes show. Before answering, check
that the letter you give is the better solution's, not the one you found at fault.
Answer with one JSON object and nothing else. Fill its fields in this order:
- `failures`: one entry per assertion that a solution does not pass (FAIL, ERROR, TIMEOUT,
or CRASH): `assertion` (its number), `solution` (`A` or `B`), `expectation_correct` (whether
the assertion's expectation is right for the task), `scope` (what the failure shows about
that solution and how much of the task it affects: `misreads_task` (a misreading of the task
or an approach that cannot work), `broad` (a broad class of ordinary inputs handled
wrongly), `edge_case` (a narrow edge case), or `none` (the expectation is wrong, so the
failure shows nothing)), and `what_it_shows` (a sentence).
- `most_serious`: for each of `A` and `B`, the most serious flaw its failures show, or that
none was shown.
- `weighing`: which solution's flaws matter more for the task as described. The number of
assertions on each side is not the measure: one failure that shows a solution misreads the
task outweighs several that show edge cases, and an assertion with a wrong expectation
counts for nothing.
- `verdict`: `A`, `B`, or `UNCERTAIN`, the better solution, decided from the fields above.
```

B.2.3 CONTROL PROMPTS

The 6 control prompts. The first, plain, is shown in full, with the task and the two solutions replaced by placeholders; every other prompt variant is given as its difference from the closest earlier variant: the sentences and lines it drops are marked − in red, the ones it adds + in green, in the order they appear.

plain

```
[system prompt]
You are a meticulous code auditor. You surface flaws in code with test assertions, and you
never claim a flaw that is not there.
[user message]
You are comparing two Python solutions, A and B, to the same programming task. Your goal is
to surface behaviour that shows which solution does what the task asks better. You do not
need to solve the task yourself, trace the programs line by line, or work out where the
problem comes from: what matters is finding inputs on which A and B could give different
outputs, and knowing the right output for them.
A separate judge decides which solution is better, or answers UNCERTAIN if it cannot tell.
The judge sees only the task description, the assertions, and each assertion's outcome on
solution A and on solution B (PASS, FAIL, ERROR with the exception type, or TIMEOUT). The
judge never sees the code. It answers UNCERTAIN unless the assertions distinguish the
solutions, and it checks each assertion against the task description, so an assertion whose
expectation is wrong will not help.
Rules for assertions:
- The block holds only plain Python `assert` statements, one per line, inside a single
```python block, and nothing else: no `import`, `def`, `class`, `exec`, assignment, `print`,
comment, or prose. A block containing any of these is rejected, and so is one that defines,
copies, or reimplements the code under review.
- Each assertion runs on its own against each solution, with the function named in the task
already defined (do not define it yourself) and these modules available: math, re,
itertools, collections, functools, string, heapq, bisect, operator, fractions, decimal,
statistics, copy, json, typing. An assertion that does not finish within 2 seconds is
reported as TIMEOUT.
- Prefer assertions whose outcomes differ between the two solutions and whose expectation is
clearly right for the task.
- At most 24 assertions in all, including previously proposed assertions; a larger set is
rejected.
```
```

```
## The case
### Task
<task description: the function's signature and docstring>
### Solution A
```python
<solution A>
```
Solution B
```python
<solution B>
```
After your reasoning, output only the code block, with no text before or after it:
```python
assert ...
```
```

minimal: as **plain**, with

```
[system prompt]
- You surface flaws in code with test assertions, and you never claim a flaw that is not
there.
+ You surface flaws in code with the smallest possible set of test assertions, and you never
claim a flaw that is not there.
[user message]
- Your goal is to surface behaviour that shows which solution does what the task asks
better.
+ Your goal is to surface behaviour that shows which solution does what the task asks
better, using as little test code as possible.
```

lean: as **plain**, with

```
[system prompt]
- You surface flaws in code with test assertions, and you never claim a flaw that is not
there.
+ You surface flaws in code with the smallest possible set of test assertions: every
assertion you write is one the judge needs, and you never claim a flaw that is not there.
[user message]
- Your goal is to surface behaviour that shows which solution does what the task asks
better.
+ Your goal is to surface behaviour that shows which solution does what the task asks
better, with the smallest set of assertions that can show it: nothing the judge does not
need to tell the two apart.
+ ## The set you write
+ Write the smallest set of assertions that lets the judge tell which solution is better,
and nothing else.
+ Every assertion must earn its place:
+ - An assertion belongs in the set only if its outcome differs between A and B and that
difference bears on which solution does the task better.
+ An assertion both solutions pass, or both fail, tells the judge nothing about which is
better: leave it out, however informative it looks on its own.
```

firm: as **lean**, with

```
[user message]
+ - Do not add assertions to be thorough, to cover the task, or to show your work.
+ The judge does not reward coverage: every extra assertion above the required set is a
waste, and every assertion that does not change the judge's decision is dead weight.
+ - The only reason to keep more than one assertion is that the judge needs each of them to
reach the right decision: for instance, when both solutions are flawed, one assertion
showing each solution's flaw, so that the judge can weigh them against each other.
+ If a single assertion decides the case, the set is that one assertion.
```

strict: as **firm**, with

```
[user message]
- Every assertion must earn its place:
+ Every assertion must earn its place, and the standard is strict:
+ - Two assertions that show the same flaw on the same kind of input are one too many: keep
the clearest, drop the rest.
+ - Before you answer, go through your set once more, assertion by assertion, and remove
every one whose removal would leave the judge's decision unchanged.
+ What remains is your answer.
```

judge: as **minimal**, with

```
[user message]
+ ## What the judge looks for
+ The judge starts from the assertions whose outcomes differ between A and B: only those can
decide the case, and an assertion with the same outcome on both cannot favour either
solution.
+ It counts an outcome for or against a solution only insofar as the assertion is a correct
expectation for the task: it checks each expectation against the task description itself,
```

```
working small inputs by hand where needed, so an input and output taken from the task's own
examples are right by definition, and an assertion with a wrong expectation helps neither
solution.
+ A PASS also tells the judge what a solution does on that input, which it checks against
the task too.
+ If the evidence does not distinguish the solutions, it answers UNCERTAIN, and it never
picks a solution on a guess about code it cannot see.
+ It keeps its reasoning short: the differing outcomes usually settle the case.
+ So write the smallest set of assertions that decides it: assertions on which A and B
differ, whose expectation is clearly right for the task, and whose inputs are small enough
for the judge to verify by hand.
+ An assertion the judge cannot use this way adds nothing.
```

B.2.4 MECHANISM-LIKE PROMPTS

The 6 mechanism-like prompts: 3 for the Second Opinion and 3 for the Rebuttal.

SO-reward: as **minimal**, with

```
[user message]
+ ## Your role
+ How you are scored:
+ Setup. Two auditors work on this case, one after the other.
+ You are the first: you propose a set of assertions.
+ A second auditor, a fresh model that sees the task, the solutions, and your set, then adds
any further assertions it likes (possibly none); the final set is your set plus its
additions.
+ The judge makes separate decisions on your set and on the final set, and the scores depend
on whether its decision changes between them.
+ The decision is A, B, or UNCERTAIN; with no assertions, or with assertions whose outcomes
do not distinguish the solutions, it is UNCERTAIN.
+ Simplicity. Every assertion set has a simplicity score S.
+ S = 1 - n / 24, where n is the number of assertions in the set: the empty set scores 1,
each assertion costs 1 / 24 = 0.042, and a set of 24 assertions scores 0.
+ Your score is 2 * S(your set) - S(final set), minus 4 if the judge's decision on the final
set differs from its decision on your set (UNCERTAIN to A or B, or A to B or B to A, all
count).
+ The second auditor's score is the negative of yours.
+ An answer that does not follow the format scores -10, and your set is then empty.
+ Since the final set contains your set, this comes to S(your set), plus what the second
auditor's additions cost in simplicity, minus 4 if they change the decision: each of your
own assertions costs you 0.042, each assertion the second auditor adds earns you 0.042, and
a changed decision costs you 4, more than any set of assertions is worth.
```

SO-reward+strategy: as **SO-reward**, with

```
[user message]
+ Examples:
+ - You propose 2 assertions (S = 0.917) that make the decision A, and no further assertion
could change it.
+ The second auditor adds nothing: 2 * 0.917 - 0.917 = 0.917.
+ If it adds 1 assertion anyway, leaving the decision A: 2 * 0.917 - 0.875 = 0.958.
+ - You propose 1 assertion (S = 0.958) that makes the decision A, but a further assertion
would turn it to B.
+ The second auditor adds it: 2 * 0.958 - 0.917 - 4 = -3.000.
+ Proposing both assertions yourself, so that the decision is B and nothing changes it:
0.917.
+ - You propose the empty set (S = 1.000), so the decision is UNCERTAIN, and the second
auditor adds one assertion that makes it A: 2 * 1.000 - 0.958 - 4 = -2.958.
+ What follows for you:
+ - Propose the smallest set of assertions that leads the judge to the decision the full
evidence supports and that no further assertions could change.
+ An assertion you leave out that the second auditor can use to change the decision costs
you 4.
+ - Do not go beyond that set: each further assertion costs 0.042 and earns nothing.
```

SO-target-set: as **minimal**, with

```
[user message]
+ ## Your role
+ Setup. Two auditors work on this case, one after the other.
+ You are the first: you propose a set of assertions.
+ A second auditor, a fresh model that sees the task, the solutions, and your set, then adds
any further assertions it likes (possibly none); the final set is your set plus its
additions.
+ The judge makes separate decisions on your set and on the final set.
+ The decision is A, B, or UNCERTAIN; with no assertions, or with assertions whose outcomes
do not distinguish the solutions, it is UNCERTAIN.
+ The set we want from you has three properties, in this order of importance:
+ 1. Right: the judge's decision on your set is the decision the whole truth about the two
solutions supports, the one the judge would reach if it saw the outcome of every assertion
```

```
that could be written.
+ 2. Settled: nothing the second auditor could add to your set would change the judge's
decision on it.
+ Whatever assertions are added, the decision on the final set is the decision on yours:
your set already carries the evidence that decides the case and leaves no opening for
evidence that would overturn it.
+ 3. Smallest: among the sets with the first two properties, yours has the fewest
assertions.
+ An assertion that is not needed for the decision to be right and to stay settled is left
out.
+ A set that is right but not settled is not what we want, and neither is one that is
settled but not right.
+ Smallness comes last: never drop an assertion the first two properties need.
```

R-reward: as **minimal**, with

```
[user message]
+ ## Your role
+ How you are scored:
+ Setup. Two auditors work on this case in three rounds.
+ Round 1: you propose a set of assertions.
+ Round 2: a second auditor, a fresh model that sees the task, the solutions, and your set,
adds any further assertions it likes (possibly none).
+ Round 3: you see your set and the second auditor's additions and add any further
assertions you like (possibly none); if the second auditor added nothing, there is no round
3 and your set is final.
+ Each set contains the one before it.
+ The judge makes separate decisions on the first set and on the final set, and the scores
depend on whether its decision changes between them; its decision on the second set does not
enter the scores.
+ The decision is A, B, or UNCERTAIN; with no assertions, or with assertions whose outcomes
do not distinguish the solutions, it is UNCERTAIN.
+ Simplicity. Every assertion set has a simplicity score S.
+ S = 1 - n / 24, where n is the number of assertions in the set: the empty set scores 1,
each assertion costs 1 / 24 = 0.042, and a set of 24 assertions scores 0.
+ Your score is 4 * S(set 1) - 2 * S(set 2) + S(set 3), minus 8 if the judge's decision on
the final set (round 3) differs from its decision on your first set (round 1) (UNCERTAIN to
A or B, or A to B or B to A, all count).
+ The second auditor's score is the negative of yours.
+ An answer that does not follow the format scores -10; in round 1 your set is then empty,
in round 3 you then add nothing.
+ Since each set contains the one before, this comes to 3 * S(set 1), plus 0.042 for each
assertion the second auditor adds, minus 0.042 for each assertion you add in round 3, minus
8 if the final decision differs from the decision on your first set.
+ So each assertion in your first set costs you 0.125, and the decision is worth 8.
+ The second auditor gains only if its change of the decision survives your round 3, so it
will add assertions when it can change the decision and you cannot change it back.
```

R-reward+strategy: as **R-reward**, with

```
[user message]
+ Examples:
+ - You propose 2 assertions (S = 0.917) that lead to the decision A, which the full
evidence supports.
+ The second auditor adds nothing and so do you: 3 * 0.917 = 2.750.
+ - You propose 1 assertion (S = 0.958) that leads to the decision A, but the full evidence
supports B.
+ The second auditor adds 1 assertion that turns the decision to B (S = 0.917), and nothing
you add in round 3 turns it back: 4 * 0.958 - 2 * 0.917 + 0.917 - 8 = -5.083.
+ Proposing both assertions in round 1 instead: 2.750.
+ - You propose 1 assertion (S = 0.958) that leads to the decision A, which the full
evidence supports.
+ The second auditor adds 1 assertion that turns the decision to B anyway, and in round 3
you add 1 assertion that turns it back to A (S = 0.875): 4 * 0.958 - 2 * 0.917 + 0.875 =
2.875.
+ Adding nothing in round 3 instead: 4 * 0.958 - 2 * 0.917 + 0.917 - 8 = -5.083.
+ What follows for you:
+ - Propose the smallest set of assertions that leads the judge to the decision the full
evidence supports, even if the second auditor could add assertions that change the judge's
decision: in round 3 you can add assertions that change it back.
+ A final decision different from the one your set leads to costs 8; each assertion costs
0.125.
+ - In round 3, if the second auditor's additions changed the judge's decision and further
assertions would restore it, add the smallest such set; otherwise add nothing.
```

R-target-set: as **minimal**, with

```
[user message]
+ ## Your role
+ Setup. Two auditors work on this case in three rounds.
+ Round 1: you propose a set of assertions.
+ Round 2: a second auditor, a fresh model that sees the task, the solutions, and your set,
adds any further assertions it likes (possibly none).
```

```
+ Round 3: you see your set and the second auditor's additions and add any further
assertions you like (possibly none); if the second auditor added nothing, there is no round
3 and your set is final.
+ Each set contains the one before it.
+ The judge makes separate decisions on your set and on the final set.
+ The decision is A, B, or UNCERTAIN; with no assertions, or with assertions whose outcomes
do not distinguish the solutions, it is UNCERTAIN.
+ The set we want from you has two properties, in this order of importance:
+ 1. Right: the judge's decision on your set is the decision the whole truth about the two
solutions supports, the one the judge would reach if it saw the outcome of every assertion
that could be written.
+ 2. Smallest: among the right sets, yours has the fewest assertions.
+ An assertion that is not needed for the decision to be right is left out.
+ That is all.
```

B.2.5 LLM TRAINING

Setting. As above, we use solutions to HumanEval problems from the EvalPlus dataset (Liu et al., 2023) and applied the same filters. We took a maximum of eight correct solutions and twenty-four incorrect solutions per task and generated two sets of solution pairs: correct-vs-buggy and buggy-vs-buggy. This gave us 835 correct-vs-buggy pairs and 836 buggy-vs-buggy pairs before splitting into train and test sets by problem. During training we sampled evenly from the correct-vs-buggy and buggy-vs-buggy pools.

We capped the number of allowable test cases at 8 and used $S(n) = 1 - \frac{n}{8}$ (where n is the number of tests) as the simplicity function. In principle, models could have hacked this scoring function by combining multiple test cases into one statement, but in our experiments simple instructions sufficed to prevent it. Models were prompted with a description of the memoryless second opinion mechanism, the problem description, the two code samples, and the previous turn's list of assertions, and instructed to respond with an action – "done", "add", or "new" – and a Python code block with a list of assertions. Like in the MNIST setting, reporting "done" ends the game with a repetition of the previous set of assertions, reporting "add" combines the listed assertions with the previous set of assertions, and reporting "new" creates a new set of assertions with only those listed by the turn agent. The agents were not provided with the outcomes of either their own test cases or the previous set of test cases. Malformed responses, including those that exceeding the reasoning budget of 2048 tokens, were given a fixed score of -4.

To start each game, we generated an initial set of assertions to represent Π_0 by sampling between 0 and 8 assertions from the EvalPlus test cases. These had a 25% chance of being a test that resolved the same on both solutions, a 12.5% for each solution of favoring them, and a 50% chance to be otherwise randomly drawn, repeating the process if a test of the desired type was unavailable.

Training. We trained rank-16 LoRAs on Granite 4.2 (3B, 8B, and 30B) (Granite Team, IBM, 2026) Nemotron 3 (30B-A3B, 120B-A12B, and 550B-A55B) (NVIDIA, 2026), and Qwen (3.6 35B-A3B and 3.8 27B) (Alibaba Cloud, 2026) using Dr. GRPO (Liu et al., 2025).

Judge. For each model, the judge was a copy of it with reasoning turned off. This copy was also trained with a rank-16 LoRA, to match the distribution of responses that it would output with reasoning turned on, on a held out set of comparisons. The judge model was prompted with the problem description, the test cases provided by the models, and their outcomes on each of the code samples (assertions were classified as `PASS`, `FAIL`, `TIMEOUT`, `ERROR` with the error type provided, or `CRASH` indicating that the process running the assertion died). It was also given access to the code of the two solutions. The judge was prompted to pick the better solution or to report `UNCERTAIN` if it could not determine which was better.

We sampled the logits of the judge twice, with each order of the solutions reversed in half of them. A report was treated as changing the judge's decision if it shifted the margin by at least $\delta = 0.5$.

Policy. Models were told to start their response by choosing exactly one action:

- *done*: repeat the previous partition and end the game

- *add*: Add additional tests to the existing suite

- *new*: Provide a new set of tests

Models that chose "add" or "new" then wrote their set of tests.

Game and reward. Agents receive the memoryless second opinion score. The simplicity and bonus terms are computed exactly with judge queries; the penalty term $\mathbb{1}_{i,i+1}[2+S(\Pi_i)+S(\Pi_{i+1})]$ is estimated by sampling 4 replies to Π_i from the current policy and taking their mean. If the game ends on the 5th turn, rather than by triggering the usual termination rule, we sample the $\mathbb{1}_{i,i+1}$ penalty term as if there were a subsequent turn so that ending the game early does not impact the reward structure.

Results In addition to the training curves provided in the main body of the paper, we conducted an evaluation on a held out set of 33 questions, over which we had 186 correct vs. buggy and 182 buggy vs. buggy pairs.

Table 4: Held-out evaluation of each agent before and after training

	Tests per report		% correct, clear pairs		% malformed		Rounds per game	
Model	Start	End	Start	End	Start	End	Start	End
Granite 4.2 3B	4.0	5.3	84	78	1.9	1.1	2.5	2.8
Granite 4.2 8B	5.4	2.4	95	87	10.9	0.2	2.3	2.6
Granite 4.2 30B	3.7	3.1	90	88	1.7	2.9	2.2	2.5
Nemotron 3 Nano	3.9	1.4	94	85	93.8	0.2	1.0	2.3
Nemotron 3 Super	2.8	2.9	89	90	2.8	1.2	2.6	3.1
Nemotron 3 Ultra	2.5	2.6	95	92	24.4	0.4	1.9	3.1
Qwen 3.6 35B-A3B	4.0	1.8	95	89	92.4	0.7	1.0	1.9
Qwen 3.8 27B	2.8	2.4	96	90	31.8	0.6	1.6	2.3