When to Overrule the Majority Vote? Precision Under Disagreement in Test-Time Reasoning

Anonymous authors
Paper under double-blind review

Abstract

LLM-as-a-judge, re-solving, and other correction mechanisms are often used to improve self-consistency by overruling its majority vote. These methods are usually judged by overall accuracy, which is a crude measure because a net gain can hide the right answers they break behind the wrong ones they fix. We instead measure a method's precision under disagreement, meaning how often it is right when it overrules the vote, and compare it with how often the overruled vote was right on those same questions. We study three open reasoning models on competition math, focusing on questions where their samples split. On the test contests, the correct answer is among the five most common answers for only 38% of wrong votes, so choosing among candidates has limited headroom. Unanimous same-model verification is 79.5% precise overall but right on only 17% of its overrides, while the votes it overruled were right on 50% of them. Used alone, the model's own executed programs fix about as many answers as they break, but when they all agree on a different answer, they are almost always right. Overriding the vote only in that case raises test accuracy by 1.6 points and breaks a right answer only once across all contests.

1 Introduction

Self-consistency samples many solutions to a problem and returns the most common final answer (Wang et al., 2022). When a model is consistently wrong, its samples agree on the wrong answer, and more samples only make the wrong vote more confident. One solution is to let another source of evidence overrule the vote. Some examples are a judge can pick another candidate (Chen et al., 2023c; Toshniwal et al., 2025), a learned aggregator can reweight the samples (Yin et al., 2024; Zhao et al., 2025), a verifier can reject the vote's answer, and other models or executed programs can offer an answer of their own.

These mechanisms are usually scored by the accuracy of the corrected output, or by how accurate the evidence source is by itself. Accuracy adds up two opposite effects. Fixing ten wrong votes while breaking eight right ones earns the same score as fixing two and breaking none, although someone who cannot check which answers changed would want only the second. Huang et al. (2024) found that models asked to correct themselves often make their answers worse. We ask, when does overruling the vote help and how often does it break a right answer?

We measure an override on the questions where it changes the answer. On each of these the source and the vote give different answers, so at most one is right, and the change is a rescue, a regression, or a swap of one wrong answer for another. The net effect equals the number of overrides times the difference between the source's precision on them (π_{dis}) and the precision the overruled vote had on the same questions (ρ_{dis}). A source right on more than half of its overrides always helps. Most sources we study fall below that, and then the vote's precision matters as much as the source's: at 30% precision, a source helps against votes that were right 10% of the time on those questions and hurts against votes that were right half the time. A verifier's overall precision also counts the many questions on which it agrees with the vote, so it cannot show this.

We study three open reasoning models on nine competition-mathematics contests and held back five more contests until every rule was fixed. On the test contests, most wrong votes do not have

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.

the correct answer among their five leading candidates, which limits any method that selects among candidates. Same-model verification and re-solving are right on few of the votes they overrule. Once gated, a much larger judge breaks almost as many answers as it fixes on the stronger models' pools, and rules with similar accuracy gains break right answers at very different rates. Programs written by the model and run by an interpreter behave differently. By themselves they are about as accurate as the vote, but when all of them agree against it we rarely see them wrong. We pre-registered the candidate-restricted version of this rule, and the open version, chosen later on one model, points the same way on the models and contests that were not used to choose it.

Contributions.

1. An analysis of overrides of a majority vote by their rescues and regressions, with exact bounds, and by the source's precision on its overrides next to the vote's precision on the same questions (Section 2), extending the flip counts used in self-correction and deferral work (Huang et al., 2024; Mozannar & Sontag, 2020) to held-out regressions and to rules with different coverage.
2. A comparison of same-model, executed and cross-model evidence, published combination rules, stronger judges and a process reward model in these terms (Sections 4.2–4.3).
3. Evidence from rules and models fixed in advance that a strict program-agreement gate seldom breaks a right answer. Each held-out comparison has at most one worsened question, and the exact upper bounds on the regression rate are about 25% (Section 4.4).

2 Measuring an Override

2.1 Votes, candidates, and contested pools

For each question a model samples reasoning traces that end in a final answer; traces that are truncated or give no answer abstain. Answers are grouped by mathematical equivalence, the majority vote $\hat{a}$ is the largest group, and the candidates $\mathcal{C}_K(x)$ are the K largest groups ($K = 5$). When the vote is wrong, the correct answer may be a candidate, appear elsewhere in the pool, or not appear at all. A selector that chooses among the candidates can only fix the first case. We therefore report two oracles: *top-K*, correct when some candidate is correct, and *pool*, correct when some trace is. All rules act only on *contested* pools, where the vote is least settled (Section 3); elsewhere the vote stands.

2.2 Override rules

An evidence source returns n answers for a question, for example the printed outputs of eight executed programs. Let s be the most common of them and σ its share of the answering samples. For a threshold θ, the confidence event U_θ holds when at least m samples answer, $\sigma \geq \theta$, and no other answer ties with s. An *override rule* outputs s when U_θ holds and s differs from the vote, and keeps the vote otherwise; at $\theta = 1$ all answering samples must agree. A rule is *open* if s may be any answer and *candidate-restricted* if it may only switch to one of the five candidates. We use $\theta \in \{0.5, 0.75, 1\}$ with $m = 6$ for eight samples.

The rule we study most closely uses the model's own programs at $\theta = 1$: on a contested pool it overrules the vote only if at least six of eight programs run to an answer, all of them agree, and the answer differs from the vote. Before any result, our internal written protocol named the candidate-restricted version of this rule as its primary rule; we call it the pre-registered rule. The open version was chosen later on the R1-8B pools and fixed before the other two models were run (Appendix C).

2.3 Precision under disagreement

An override is a *rescue* if the source is right, a *regression* if the vote was right, and neutral if both are wrong. Over n_{dis} overrides,

$$\text{rescues} - \text{regressions} \;=\; n_{\text{dis}}\,(\pi_{\text{dis}} - \rho_{\text{dis}}), \tag{1}$$

where π_{dis} is the share of overrides on which the source is right and ρ_{dis} the share on which the overruled vote was right. A rule helps exactly when the *disagreement advantage* $A_{\text{dis}} = \pi_{\text{dis}} - \rho_{\text{dis}}$ is positive, and ρ_{dis} is the rate at which its overrides break a right answer. Because at most one of the two answers is right, $\pi_{\text{dis}} + \rho_{\text{dis}} \leq 1$. A source right on more than half of its overrides therefore always helps; below that, ρ_{dis} decides the sign, and $1 - \pi_{\text{dis}} - \rho_{\text{dis}}$ is the share of neutral changes. Both are measured on the same questions, so ρ_{dis} also tells us how hard those questions were for the vote.

Verifiers are usually reported by their overall precision on the answers they are confident about, which mixes agreements with disagreements,

$$P(s \text{ correct} \mid U_\theta) \;=\; (1 - f_{\text{dis}})\,\alpha_{\text{agr}} \;+\; f_{\text{dis}}\,\pi_{\text{dis}}, \tag{2}$$

where f_{dis} is the share of confident answers that disagree with the vote and α_{agr} is the vote's accuracy where they agree. Agreements are usually the majority, so a verifier can look precise overall while being wrong most of the time it actually changes an answer.

2.4 Where a gain comes from

We also ask why a source helps, whether because it is more accurate than the vote overall or because its confidence is high where the vote fails. Let V and S indicate that the vote and the source's answer are correct on a random contested pool ($S = V$ when the source abstains), let g indicate that the gate fires, and let p, q and c be the vote's accuracy, the source's accuracy alone and the gate's coverage. The gated rule's correctness is $R = gS + (1 - g)V$, so

$$\mathbb{E}[R] - \mathbb{E}[V] \;=\; \underbrace{c\,(q - p)}_{\text{accuracy gap}} \;+\; \underbrace{\operatorname{Cov}(g,\, S - V)}_{\text{selectivity}}. \tag{3}$$

This is an elementary identity, the accounting of learning to defer (Mozannar & Sontag, 2020) and of human–algorithm teams (Donahue et al., 2022). We use it for one point. A source no more accurate than the vote can still help if its confidence is high on the questions where the vote is wrong, and being external to the model or having errors unrelated to the vote's does not by itself produce a gain. How the gain splits between the two terms depends on how the source alone is scored, so we use the split only descriptively (Appendix P); Eq. (1) needs no such choice.

3 Experimental Setup

Models and pools. We use DeepSeek-R1-0528-Qwen3-8B ("R1-8B"), gpt-oss-20b and Qwen3-30B-A3B-Thinking-2507 ("Qwen3-30B"), sampled at temperature 0.6; the gpt-oss models ran at their chat template's default (medium) reasoning level (Appendix A). Each question first receives 16 traces. If the leading answer holds at least 80% of them the vote is kept; otherwise sampling continues to 64 traces, and the pool is *contested* if the leading answer still holds less than 80%. Evidence sources are run, and rules applied, only on contested pools. gpt-oss-20b and Qwen3-30B have three independent runs per question and R1-8B one to six, some of which share most of their traces (Appendix A). Because runs of the same question share evidence, each (model, question) pair counts once, with its change averaged over runs.

Contests. Nine MathArena contests (Balunović et al., 2025) form a development split (AIME 2024, HMMT February 2025, BRUMO 2025, SMT 2025, CMIMC 2025; 183 questions) and a test split (AIME 2025, HMMT November 2025, HMMT February 2026, AIME 2026; 123 questions, 120 for R1-8B, the last three released after every model). Five further contests (HMMT February 2023 and 2024, AIME 2023 I and II, ArXivMath August 2026; 147 questions) were run once, after every rule was fixed, as a confirmatory set. GPQA Diamond serves as a multiple-choice control (Appendix N).

Evidence sources. Table 1 lists the sources. Same-model sources reuse the base model: it resolves the problem while shown the candidates, or judges each candidate correct or incorrect. Executed sources have a model write a Python program whose printed output is the answer; tool-integrated traces run code while reasoning. Cross-model sources use the other models' programs or traces. We also run published combination rules, stronger judges and a process reward model (Section 4.3).

Table 1: Auxiliary evidence sources used in the experiments (details in Appendix B).

Class	Source	Evidence
Same model	Candidate-conditioned re-solve	8 solutions with candidates shown
	Candidate verification	4 judgments per candidate
Executed computation	Own executed programs	8 externally executed programs
	Candidate-conditioned programs	8 programs with candidates shown
Cross model	Other-model executed programs	8 programs from another model
	Other-model reasoning pool	64 traces from another model
	Other-model reasoning (8 traces)	4 traces from each other model

Table 2: Majority-vote accuracy and candidate-availability oracle headroom (accuracies in %; last column: wrong contested votes whose answer is a top-5 candidate / all wrong contested votes).

model	split	pools	contested	vote	top-5 oracle	pool oracle	top-5 present / wrong
R1-8B	dev	183	66	78.7	88.0	95.1	15 / 33
R1-8B	test	443	151	70.9	81.5	84.9	43 / 101
gpt-oss-20b	dev	549	220	87.1	94.0	96.4	36 / 66
gpt-oss-20b	test	369	142	84.6	89.4	91.1	17 / 50
Qwen3-30B	dev	549	122	87.2	92.7	95.1	29 / 52
Qwen3-30B	test	369	88	85.9	89.4	91.1	12 / 37

Statistics. We report whole-benchmark gains averaged over runs, with bootstrap intervals that resample questions; exact sign tests over distinct questions; and exact binomial bounds on regression rates. Overrides are few, so we give the bound alongside every observed zero. The protocol and its history are in Appendix C.

4 Results

4.1 Many wrong test votes lack the answer among the candidates

Table 2 shows how often a wrong vote could be fixed at all. On the test contests the correct answer is among the five leading candidates for only 38% of wrong contested votes (40% [26, 56] when each model–question pair counts once), and appears anywhere in the pool for 50%; showing more candidates adds little (Appendix F). A judge or verifier that chooses among the candidates cannot reach most test failures however good it is, while an open source can output an answer that no trace produced. On the easier development contests the correct answer is a candidate for 53% of wrong contested votes and appears in the pool for 78%, so selection has more room there.

4.2 Same-model evidence when it disagrees with the vote

Table 3 compares each source's precision on its overrides with the vote's precision on the same questions. On development, candidate verification at $\theta = 0.5$ is right on 16% of the votes it overrules, while those votes were right 63% of the time, and a stricter margin rule does no better (6% against 59%). Re-solving the problem while shown the candidates is right on 21% of its test overrides at $\theta = 0.5$, against 12% for the overruled votes, and on 29% against 31% on development, so on the easier contests it replaces right and wrong votes at about the same rate. On the harder test contests the same rules look better ($A_{\text{dis}} = +20$ for verification), but only because the votes they overrule are rarely right there; the gain changes sign when a single contest is dropped, which happens for the program and cross-model sources only in one cell with three or four pairs (Appendix P). The strength of the vote does not explain the flip either. Among pools whose vote holds less than half of the traces, verification at $\theta = 0.5$ has $A_{\text{dis}} = +39$ on test, improving 8 pairs and worsening 2, and $A_{\text{dis}} = -62$ on development, improving 1 and worsening 10 (Table 25).

Verification at $\theta = 1$ and candidate-conditioned programs at $\theta = 0.5$ are both about 79% precise on the answers they are confident about, yet when they contradict the vote they are right 17% and

Table 3: Precision under disagreement. Each (model, question) pair counts once, its runs averaged (contested pools, three models). Overrides: pools on which the rule changes the answer, with (model, question) pairs in parentheses; * fewer than 10 pairs. π_{dis} and ρ_{dis} in %, with exact 95% intervals for π_{dis}; $A_{\text{dis}} = \pi_{\text{dis}} - \rho_{\text{dis}}$ in points. Margin: the vote's candidate rejected by ≥ 2 verdicts and one other candidate accepted by ≥ 2. More thresholds and pool-level counts: Appendices E and P.

		test				development		
source	rule	overrides	π_{dis}	ρ_{dis}	A_{dis}	π_{dis}	ρ_{dis}	A_{dis}
candidate verification	$\theta = 0.5$	48 (23)	38 [20, 61]	18	+20	16	63	−47
	margin	71 (25)	26 [12, 49]	20	+6	6	59	−53
	$\theta = 1$	7 (5)*	18 [1, 72]	55	−36	43	43	0
cand.-conditioned re-solving	$\theta = 0.5$	68 (24)	21 [10, 47]	12	+9	29	31	−2
own programs	$\theta = 0.5$	54 (17)	60 [33, 82]	18	+42	66	14	+53
	$\theta = 0.75$	37 (11)	86 [48, 98]	0	+86	74	9	+65
	$\theta = 1$	25 (8)*	95 [47, 100]	0	+95	88	4	+84
tool-integrated traces (gpt-oss-20b)	$\theta = 0.75$	12 (4)*	100 [40, 100]	0	+100	100	0	+100
other models' programs	$\theta = 0.75$	45 (15)	97 [68, 100]	0	+97	80	9	+70
other models' pools	$\theta = 0.75$	17 (10)	65 [35, 93]	0	+65	79	18	+62
8 traces, two other models	$\theta = 0.75$	19 (13)	69 [39, 91]	0	+69	93	7	+86

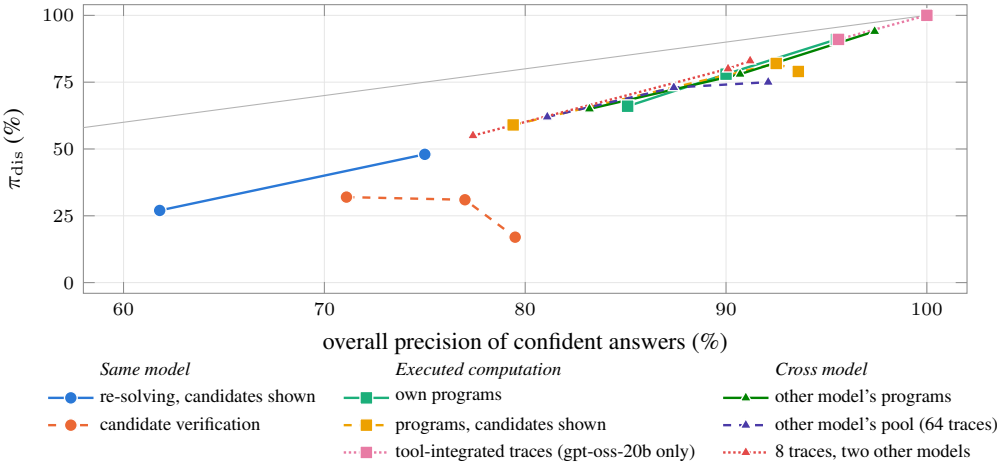


Figure 1: Overall precision of a source's confident answers against its precision when it contradicts the vote (π_{dis}), contested pools of both splits, three models pooled. Marker shape gives the class of source; color and line style give the method. One point per threshold $\theta \in \{0.5, 0.75, 1\}$, joined per method; overall precision rises with θ for every method, so each line runs left to right. Points with fewer than five disagreements are omitted. Gray line: π_{dis} equal to overall precision; pool level (Table 17).

59% of the time (Eq. 2). In Figure 1, overall precision rises with the threshold for every source. The executed and cross-model sources stay fairly precise when they disagree, while the same-model sources fall far below the diagonal. The verifier is most confident on easy questions, where the vote is already right, so its confidence says little about where the vote fails (Appendix P). Same-model sources also tend to confirm wrong votes. When verification at $\theta = 1$ or re-solving at $\theta = 0.5$ agrees with the vote, the vote is still wrong in 15% and 26% of cases, against 3% for the unanimous programs.

Table 4: Regressions pooled over every split on which a rule was run: (model, question) units improved / worsened, the worsened share of changed units (%, exact 95% interval), and the cost-weighted value, improved minus λ times worsened, where λ is the cost of breaking a right answer relative to fixing a wrong one. At $\lambda = 1$ this ranks rules by accuracy; totals are comparable only between rules run on the same splits, and the worsened share is comparable across all. With an independent block of programs per run, the $\theta = 0.75$ program override is 17/2 on development and test (gpt-oss-20b and Qwen3-30B). Extra samples per contested question beyond the model's own pool; eight programs cost 5–8% of a run's output tokens, the jury three times the sampling. Counted by distinct question (net change over its models): 24/2 (upper bound 25.1%) at $\theta = 0.75$, 19/1 (24.9%) at $\theta = 1$, 23/2 (26.0%) for other models' programs.

rule	splits	extra samples	+/−	worsened share	$\lambda = 1$	$\lambda = 3$	$\lambda = 10$
programs, open, $\theta = 0.75$	dev, test, conf.	8 programs	33/2	5.7 [0.7, 19.2]	+31	+27	+13
programs, open, $\theta = 1$	dev, test, conf.	8 programs	22/1	4.3 [0.1, 21.9]	+21	+19	+12
programs, candidates, $\theta = 1$ (pre-reg.)	dev, test, conf.	8 programs	13/0	0.0 [0.0, 24.7]	+13	+13	+13
tool-integrated traces alone (gpt-oss-20b)	dev, test	8 TIR traces	24/1	4.0 [0.1, 20.4]	+23	+21	+14
other models' programs, $\theta = 0.75$	dev, test	8 programs	30/2	6.2 [0.8, 20.8]	+28	+24	+10
other models' pools, $\theta = 0.75$	dev, test	64 traces	18/1	5.3 [0.1, 26.0]	+17	+15	+8
8 traces, two other models, $\theta = 0.75$	dev, test	8 traces	21/1	4.5 [0.1, 22.8]	+20	+18	+11
3-model jury, majority of models	dev, test	128 traces	36/4	10.0 [2.8, 23.7]	+32	+24	−4
CoT–PoT voting, counts	dev, test	8 programs	15/2	11.8 [1.5, 36.4]	+13	+9	−5
programs, open, $\theta = 0.5$	dev, test, conf.	8 programs	40/12	23.1 [12.5, 36.8]	+28	+4	−80
3-model jury, summed shares	dev, test	128 traces	56/18	24.3 [15.1, 35.7]	+38	+2	−124
CoT–PoT voting, shares	dev, test, conf.	8 programs	61/29	32.2 [22.8, 42.9]	+32	−26	−229
programs alone	dev, test, conf.	8 programs	53/45	45.9 [35.8, 56.3]	+8	−82	−397

4.3 Accuracy gains hide regressions

The model's own programs are external to the model in the sense that matters, since an interpreter fixes their output, and their errors are less tied to the vote's than those of same-model sources (Appendix G). Used alone, replacing the vote whenever the programs give an answer, they still change test accuracy by only -0.9 points $[-3.9, +1.9]$, fixing about as many answers as they break.

Rules with similar accuracy differ in the same way. Pooled over all splits (Table 4), program and cross-model overrides with a strict gate break 0–6% of the answers they change. Chain-of-thought–program voting with share weights (Saparkhan et al., 2026) gains about as much, but 32% of its changes break a right answer. A three-model jury (Liu, 2026) by summed answer shares breaks 24%, and programs used alone 46%. Neither combination rule has a gate, so its whole gain comes from being a more accurate predictor than the vote, and that gain depends on the base model: the jury by summed shares gains $+7.0$ points on R1-8B's test pools but between $+0.5$ and $+1.8$ on those of gpt-oss-20b and Qwen3-30B (Appendix P). Tool-integrated traces, which we could run only for gpt-oss-20b, are more accurate than the vote on their own ($+4.1$ points on its test units) and were right on all 12 of their test overrides at $\theta = 0.75$ (Table 3). When breaking a right answer costs no more than fixing a wrong one, the looser rules can come out ahead; when it costs more, for example because changed answers cannot be re-checked, the strict gates lead. Across all 32 rules, the number of right answers a rule breaks on development predicts the number it breaks on test (Spearman 0.88, partly because both grow with how often the rule acts), whereas its development gain does not detectably predict the share of its changes that break a right answer (Appendix P).

Where the gains come from. Splitting the gains with Eq. (3) under the convention of Appendix P (Table 21, Figure 3), the model's own programs have an accuracy gap close to zero at every threshold, and their gain is selectivity: $+1.6$ $[+0.4, +2.9]$ points on test for the unanimous rule and $+1.0$ $[+0.4, +1.7]$ on development. Tool-integrated traces are more accurate than the vote, and more than half of their $\theta = 0.75$ gain, $+1.9$ $[+0.4, +4.1]$ of $+3.3$ points, is accuracy gap. The eight traces from two other models gain through selectivity ($+0.8$ $[+0.1, +1.5]$ on test), and for the same-model sources both terms stay within 0.4 points of zero. With 16 to 64 programs, programs alone become more accurate than the vote and part of their gain moves into the gap, while their precision under disagreement stays high.

Table 5: A much larger judge against the model's own programs on the same pools: all contested pools of gpt-oss-20b and Qwen3-30B on development, test and confirmatory contests (721 pools). Distinct questions improved and worsened (net change over models and runs), exact two-sided sign test, and precision on the overrides (π_{dis}) against the overruled vote's (ρ_{dis}; for the programs see Table 3). The judge is GenSelect-style selection by gpt-oss-120b (eight samples, candidates shown).

source	rule	improved	worsened	p	π_{dis}	ρ_{dis}
gpt-oss-120b judge	alone	23	35	0.15	24	34
	$\theta = 0.75$	14	11	0.69	42	30
	$\theta = 1$	5	2	0.45	55	25
own programs, candidates only	$\theta = 0.75$	14	0	0.0001	–	–
	$\theta = 1$	11	0	0.001	–	–
own programs, open	$\theta = 0.75$	18	1	0.0001	–	–
	$\theta = 1$	14	1	0.001	–	–

Table 6: Evidence that does not depend on choices made after seeing the data: distinct questions improved and worsened by the program override (net change over models and runs), with two-sided exact sign tests. Open: the rule may output any answer; candidates: only one of the five leading candidates.

evidence	rule	improved	worsened	p
pre-registered rule, all contests and models	candidates, $\theta = 1$	13	0	0.0002
models run after the rule was fixed (gpt-oss-20b, Qwen3-30B), problems rescued for R1-8B excluded	open, $\theta = 1$	12	1	0.003
five confirmatory contests, run once at the end	open, $\theta = 1$	5	0	0.06
	open, $\theta = 0.75$	7	0	0.016 (Holm 0.13)
three test contests released after the models	open, $\theta = 1$	4	0	0.13

Stronger judges. We also tested whether a stronger judge would do better than the model's own evidence. We ran GenSelect-style selection (Toshniwal et al., 2025), in which a model re-solves the problem while shown the candidates, with gpt-oss-20b, Qwen3-30B and the much larger gpt-oss-120b as judges, as well as a process reward model (Zhang et al., 2025) (Table 24). On R1-8B's pools, the weakest base model, gpt-oss-120b helps: gated at $\theta = 0.75$ it gains $+4.9$ points on test [$+1.7$, $+8.9$], comparable to R1-8B's own programs. The judges of the same size as the base models do less on these pools: gated at $\theta = 0.75$, GenSelect-style selection gains $+2.3$ points with gpt-oss-20b as the judge and $+1.1$ with Qwen3-30B, and verification by Qwen3-30B with a margin gate gains $+0.2$. On the pools of the two stronger models it does not (Table 5). The judge only chooses among the five candidates, so the fair comparison is with the programs restricted in the same way: at $\theta = 0.75$ the judge's overrides improve 14 questions and worsen 11 (right on 42% of its overrides against 30% for the overruled votes), while the candidate-restricted programs improve 14 and worsen none ($p = 0.0001$). Used alone the judge improves 23 questions and worsens 35; this difference is not significant ($p = 0.15$), and on R1-8B's development pools its gated gain is -1.1 points, so the result shows that the judge is imprecise on these pools and leaves open whether it does harm overall. The judge ran at its chat template's default reasoning level (median 8.6k tokens per sample). Being more capable did not make the judge more precise here, and its value depended on how often the overruled vote was right. The process reward model scores candidates without solving. Used alone to pick the best candidate it loses 6.1 points on test, and gated at a candidate share of 0.5 it still loses 0.9.

4.4 Executed programs: precise when they agree against the vote

The programs' precision rises steadily as the gate tightens (Table 3). When all answering programs agree against the vote, they are right on 95% of the test overrides and 88% of the development overrides, while the overruled votes were right on none and 4%. On their own the programs are about as accurate as the vote, so this precision is where their gain comes from.

Evidence fixed in advance. Table 6 lists three tests of rules that were fixed before the data they are tested on. First, the pre-registered rule, the candidate-restricted version of the unanimous program

override, improves 13 distinct questions and worsens none across all contests and models (exact sign test $p = 0.0002$). It did not meet the protocol's own criterion, which asked for significance on the test split for every model. On test it improves only 2 questions, and the pooled test above is an analysis we chose afterwards. Its gains are modest because it cannot leave the candidates, and over half of the open rule's rescues lie outside them. Second, the open rule was chosen on R1-8B and then fixed before gpt-oss-20b and Qwen3-30B were run. On those two models, leaving out every problem it had rescued for R1-8B, it improves 12 questions and worsens one ($p = 0.003$). The one regression is a question whose published answer key is contradicted by exact enumeration (Appendix R). Third, on the five confirmatory contests, run once after all analyses, the unanimous rule improves 5 questions and the $\theta = 0.75$ rule 7, and neither worsens any. With eight rules run there, the $\theta = 0.75$ result ($p = 0.016$) does not survive a Holm adjustment (0.13), so this set supports the direction of the effect but cannot establish its size. On the three test contests released after all three models, the unanimous rule improves 4 questions and worsens none. The confirmatory rescues, by contrast, come from contests that predate the models. The models do not seem to recall these answers, since on each rescued question at most 12 of the 64 traces reach the reference.

An example. AIME 2025 asks for the number of ordered triples (a, b, c) of positive integers up to 3^6 with $a^3 + b^3 + c^3$ divisible by 3^7, modulo 1000. Both R1-8B and gpt-oss-20b produce contested pools that lead with 147, and for R1-8B the correct answer, 735, is not among the pool's five leading candidates, so no judge restricted to the candidates could recover it. Each model's eight programs enumerate the residues directly, all that answer print 735, and the rule overrules the vote in every run. The traces agree on a slip in a long modular computation, and the interpreter does not make it. Same-model verification fails differently: half of its unanimous overrides rest on a single "correct" verdict for a runner-up candidate among otherwise rejected ones (Appendix B).

Where execution helps. The rescued problems are ones whose answer a short program can enumerate. Programs catch computational slips. When they share the pool's misreading of a problem, they agree with the wrong vote and the rule stays silent, as it does on geometry and on answers with radicals (Appendix Q). A keyword classifier that sends every counting question to the programs' plurality is right on 42% of its test overrides and 55% of its development overrides, against 30% and 18% for the overruled votes, far below the agreement gate, and about half of the unanimous rule's gain comes from questions the classifier does not mark (Appendix P). The result does not depend on one draw of programs. When gpt-oss-20b and Qwen3-30B receive a fresh block of eight programs for every run, the unanimous override improves 5/0 test pairs ($+0.8$ points) and the $\theta = 0.75$ override 7/1 ($+1.8$ points), so the gain shrinks and the $\theta = 0.75$ rule picks up one regression (Appendix U). Several controls tie the precision to the finished program. The answer the model states in its reasoning before writing the code is right on only 4–5% of the pools where it contradicts the vote (Table 29). Eight fresh reasoning traces at the $\theta = 0.5$ gate are right on 18% of their test overrides and 7% of their development overrides, against 27% and 75% for the overruled votes. When gpt-oss-20b predicts what its own programs print without running them, the predictions are as precise when they disagree with the vote, but they disagree on no test pool and on 9 development pools, against 15 and 27 for the executed programs, so we cannot fully separate writing a program from running it. Unanimity also depends on the budget of eight programs. With 32 or 64 programs they rarely all agree, while the $\theta = 0.75$ gate stays precise (π_{dis} 73–92%), though at 32 programs it breaks one right answer on test (Appendix P). Eight programs add 7.6–7.8% (gpt-oss-20b), 5.3–5.7% (Qwen3-30B) and 4.8% (R1-8B) to a run's output tokens (Table 34).

5 Related Work

Self-correction. Language models do not reliably correct themselves without external feedback (Huang et al., 2024); self-verification is unreliable where a sound external verifier helps (Stechly et al., 2024), and resampling against an imperfect verifier is limited by its false positives (Stroebl et al., 2024). Compute can also be divided between solving and verifying (Singhi et al., 2025). These works contrast internal and external feedback. We instead contrast what a source fixes with what it breaks.

Self-consistency, selection, and verification. Self-consistency and its adaptive and confidence-weighted variants (Wang et al., 2022; Aggarwal et al., 2023; Li et al., 2024; Fu et al., 2025;

Zhou et al., 2025) aggregate repeated samples. Generative selection, revision and learned aggregators (Chen et al., 2023c; Toshniwal et al., 2025; Yin et al., 2024; Zhang et al., 2026; Zhao et al., 2025), trained outcome and process verifiers (Cobbe et al., 2021; Zhang et al., 2025) and combinations of weak verifiers (Saad-Falcon et al., 2025) choose among candidates, whose coverage grows with sampling faster than selection can use it (Brown et al., 2024). Judges show position bias (Zheng et al., 2023; Wang et al., 2023), and self-evaluation is imperfect (Kadavath et al., 2022). Closest to our setting, ARBITER (Cai et al., 2026) targets wrong-majority failures with conservative same-model evidence and hidden-state signals, and Minority Sentinel (He et al., 2026) learns when to overturn the majority in multi-agent debate and reports correct minus wrong flips. Its flip precision leaves out overrides on which both answers are wrong, so it differs from π_{dis}. Shu (2026) give an exact identity for replacing one ballot of a vote by a verifier's verdict, which acts only on questions decided by one vote. We apply the accounting to sources outside the pool and add ρ_{dis} and held-out regressions.

Selective prediction, deferral and cascades. Equation (1) is the accounting of selective prediction, deferral and cascades (Geifman & El-Yaniv, 2017; Madras et al., 2018; Jung et al., 2025; Mozannar & Sontag, 2020). Jitkrittum et al. (2023) show that deferring on the first model's confidence falls short when the second model is a specialist; in our rules the source's own confidence is the gate. Zhao et al. (2023) analyse a selector between chain-of-thought and program-aided answers in the same terms, Guo et al. (2024) separate how often a decision-maker relies on an AI from how well that reliance is placed, and work on negative flips in model updates (Srivastava et al., 2020) and on cascades that defer on CoT–PoT agreement (Yue et al., 2024) is the closest analogue.

Programs and cross-model evidence. Program-aided and tool-integrated reasoning (Gao et al., 2023; Chen et al., 2023b; Gou et al., 2024; Moshkov et al., 2025) use execution as an alternative way to reason, while execution agreement, code-based verification and CoT–PoT ensembles use programs to filter or combine outputs (Chen et al., 2023a; Ni et al., 2023; Zhou et al., 2024; Toh et al., 2024; Saparkhan et al., 2026). Cross-model consensus and mixtures provide another source of evidence (Liu, 2026; Wang et al., 2024), although their errors remain correlated (Ding, 2026; Kohli, 2026).

6 Limitations and Conclusion

Limitations. Our evidence comes from competition mathematics and three open models of 8–30B parameters. The results rest on few disagreements. The test gain of the unanimous rule comes from five problems, and the confirmatory rescues come from contests that predate the models. Executed programs help only on problems a program can enumerate. On GPQA, same-model re-solving is right on all 13 of its unanimous overrides, so the weakness of same-model evidence may depend on the answer format. Tool-integrated reasoning as the base solver could be run for only one model, since neither Qwen3-30B nor R1-8B wrote code while reasoning when offered a tool. The gpt-oss models ran at their default (medium) reasoning level, all base models have 8–30B parameters, and on answers that a program cannot simply print, such as closed forms with radicals, the strict program rules rarely fire.

Conclusion. Across 32 override rules on competition mathematics, accuracy after correction hid how many right answers each rule broke, and the source's precision on its overrides, set next to the vote's precision on the same questions, separated the two. Same-model verification and re-solving were right on few of the votes they overruled, and a much larger judge broke nearly as many right answers as it fixed on the stronger models' pools. A model's own programs, about as accurate as its vote, overruled it with almost no regressions when they all agreed, including under a rule fixed before any result and on models and contests not used to choose it.

AI use statement

In this work, we used generative AI tools (an LLM-based coding and research assistant working under the authors' direction) to implement methods (the sampler, execution sandbox, evaluation and statistics scripts), to design and provide feedback on experiments and analyses, to propose and

refine hypotheses (including the precision-under-disagreement analysis), to interpret results, and to clean and reformat the benchmark files. We have not used generative AI tools to generate synthetic data sets, to formulate or prove mathematical claims beyond the elementary accounting identities in Section 2, or for translation, and qualitative data analysis is not applicable to this work. Additionally, we used generative AI tools to search for and summarize related literature, to format references, to suggest a structure for the paper, and to draft and edit parts of the text. We have reviewed all AI-assisted work: every number in the paper is regenerated by scripts from the stored samples, audited overrides were inspected by hand, related work was checked against the cited papers, and the authors verified the code paths that produce the reported tables. The language models whose outputs are studied (DeepSeek-R1-0528-Qwen3-8B, gpt-oss-20b, Qwen3-30B-A3B-Thinking) are objects of study, not writing tools. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

Ethics statement

This work evaluates automatic answer-correction rules on public competition-mathematics benchmarks and GPQA Diamond. It involves no human subjects, personal data or new data sets beyond reformatted public benchmarks, and we see no direct risk of harm. The rules studied change a model's answers, and the paper measures how often such changes break a right answer.

Reproducibility statement

All prompts are given verbatim in Appendix V; models, pools, splits and evidence sampling are described in Appendix A, and sampler, sandbox and grading details in Appendix W. Every sample was generated with a deterministic per-request seed and is stored with its extracted and executed answers; the evaluation scripts reproduce every table from these samples, and the code, samples and tables are provided in the supplementary material.

References

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *arXiv preprint arXiv:2110.01052*, 2021.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Meng Cai, Lars Kulik, and Farhana Choudhury. ARBITER: Reasoning trajectory basins and majority vote failures in test-time sampling. *arXiv preprint arXiv:2605.26172*, 2026.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. CodeT: Code generation with generated tests. In *International Conference on Learning Representations*, 2023a.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023b.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023c.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Kaihua Ding. When LLMs agree, are they right? auditing self-consistency and cross-model agreement as confidence signals. *arXiv preprint arXiv:2607.08065*, 2026.

Kate Donahue, Alexandra Chouldechova, and Krishnaram Kenthapadi. Human-algorithm collaboration: Achieving complementarity and avoiding unfairness. In *ACM Conference on Fairness, Accountability, and Transparency*, 2022. doi: 10.1145/3531146.3533221.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv preprint arXiv:2508.15260*, 2025.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning*, 2023.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, 2017.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *International Conference on Learning Representations*, 2024.

Ziyang Guo, Yifan Wu, Jason Hartline, and Jessica Hullman. A decision theoretic framework for measuring AI reliance. In *ACM Conference on Fairness, Accountability, and Transparency*, 2024. doi: 10.1145/3630106.3658901.

Chuan He, Zebin Chen, Zhengyi Yang, Shaobo Qiao, Mingchen Ju, Jiate Liu, Dong Wen, and Guanfeng Liu. Minority sentinel: When to overturn majority voting in multi-agent LLM debates. In *AgentSearch Workshop at SIGIR*, 2026. arXiv:2606.29270.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024.

Wittawat Jitkrittum, Neha Gupta, Aditya Krishna Menon, Harikrishna Narasimhan, Ankit Singh Rawat, and Sanjiv Kumar. When does confidence-based cascade deferral suffice? In *Advances in Neural Information Processing Systems*, 2023.

Jaehun Jung, Faeze Brahman, and Yejin Choi. Trust or escalate: LLM judges with provable guarantees for human agreement. In *International Conference on Learning Representations*, 2025.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Guneet Kohli. Nine judges, two effective votes: Correlated errors undermine LLM evaluation panels. *arXiv preprint arXiv:2605.29800*, 2026.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *International Conference on Learning Representations*, 2024.

Ning Liu. LLMs as a jury: Cross-model consensus can outperform process reward models for LLM reasoning. *arXiv preprint arXiv:2607.10139*, 2026.

David Madras, Toniann Pitassi, and Richard Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer. In *Advances in Neural Information Processing Systems*, 2018.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. AIMO-2 winning solution: Building state-of-the-art mathematical reasoning models with OpenMathReasoning dataset. *arXiv preprint arXiv:2504.16891*, 2025.

Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert. In *International Conference on Machine Learning*, pp. 7076–7087, 2020.

Ansong Ni, Srini Iyer, Dragomir Radev, Veselin Stoyanov, Wen-tau Yih, Sida Wang, and Xi Victoria Lin. Lever: Learning to verify language-to-code generation with execution. In *International Conference on Machine Learning*, pp. 26106–26128. PMLR, 2023.

Jon Saad-Falcon, E. Kelly Buchanan, Mayee F. Chen, Tzu-Heng Huang, Brendan McLaughlin, Tanvir Bhathal, Shang Zhu, Ben Athiwaratkun, Frederic Sala, Scott Linderman, Azalia Mirhoseini, and Christopher Ré. Shrinking the generation-verification gap with weak verifiers. *arXiv preprint arXiv:2506.18203*, 2025.

Raman Saparkhan, Majd Hawasly, Md Rizwan Parvez, and Mohammad Raza. Self-consistency from only two samples: CoT-PoT ensembling for efficient LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026. arXiv:2604.17433.

Yang Shu. When verifiers vote backwards under verdict substitution: Signed pivotal value in correlated self-consistency. *arXiv preprint arXiv:2609.26144*, 2026.

Nishad Singhi, Hritik Bansal, Arian Hosseini, Aditya Grover, Kai-Wei Chang, Marcus Rohrbach, and Anna Rohrbach. When to solve, when to verify: Compute-optimal problem solving and generative verification for LLM reasoning. *arXiv preprint arXiv:2504.01005*, 2025.

Megha Srivastava, Besmira Nushi, Ece Kamar, Shital Shah, and Eric Horvitz. An empirical analysis of backward compatibility in machine learning systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3272–3280, 2020.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations of large language models on reasoning and planning tasks. *arXiv preprint arXiv:2402.08115*, 2024.

Benedikt Stroebl, Sayash Kapoor, and Arvind Narayanan. The limits of inference scaling through resampling. *arXiv preprint arXiv:2411.17501*, 2024.

Vernon Y. H. Toh, Deepanway Ghosal, and Soujanya Poria. Not all votes count! programs as verifiers improve self-consistency of language models for math reasoning. *arXiv preprint arXiv:2410.12608*, 2024.

Shubham Toshniwal, Ivan Sorokin, Aleksander Ficek, Ivan Moshkov, and Igor Gitman. Genselect: A generative approach to best-of-n. *arXiv preprint arXiv:2507.17797*, 2025.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*, 2024.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*, 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Zhiyuan Zeng, Xiaonan Li, Tianxiang Sun, Cheng Chang, Qinyuan Cheng, Ding Wang, Xiaofeng Mou, Xipeng Qiu, and Xuanjing Huang. Aggregation of reasoning: A hierarchical framework for enhancing answer selection in large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, 2024.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. Large language model cascades with mixture of thought representations for cost-efficient reasoning. In *International Conference on Learning Representations*, volume 2024, pp. 21691–21728, 2024.

Yingchuan Zhang, Terry Ma, Wenxuan Zhong, and Ping Ma. A single revision step improves token-efficient llm reasoning. *arXiv preprint arXiv:2602.02828*, 2026.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, 2025.

James Xu Zhao, Yuxi Xie, Kenji Kawaguchi, Junxian He, and Michael Qizhe Xie. Automatic model selection with large language models for reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

Wenting Zhao, Pranjal Aggarwal, Swarnadeep Saha, Asli Celikyilmaz, Jason Weston, and Ilia Kulikov. The majority is not always right: Rl training for solution aggregation. *arXiv preprint arXiv:2509.06870*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, 2023.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, and Hongsheng Li. Solving challenging math word problems using GPT-4 code interpreter with code-based self-verification. In *International Conference on Learning Representations*, 2024.

Zhi Zhou, Yuhao Tan, Zenan Li, Yuan Yao, Lan-Zhe Guo, Yu-Feng Li, and Xiaoxing Ma. A theoretical study on bridging internal probability and self-consistency for llm reasoning. In *Advances in Neural Information Processing Systems*, 2025.

A Experimental and Reproducibility Details

Models and generation. We evaluate DeepSeek-R1-0528-Qwen3-8B ("R1-8B"), gpt-oss-20b at its chat template's default reasoning setting, and Qwen3-30B-A3B-Thinking-2507 (FP8 release). All samples use temperature $T = 0.6$, top-$p = 0.95$, and maximum output lengths of 60k tokens for gpt-oss-20b and 92k tokens for the other models. Truncated samples and samples without a valid boxed answer abstain.

Pool construction. Each question receives 16 warm-up traces. If the leading answer has share at least 0.8, the pool is *settled*; otherwise sampling continues to 64 traces. A 64-trace pool is *contested* when the top answer share remains below 0.8 and at least two answer clusters remain.

For R1-8B, six contests reuse pools from the earlier tournament experiments: one seed on AIME 2024, HMMT Feb 2025, and BRUMO 2025; three seeds on AIME 2025 (83 complete pools); six seeds on the 30-question version of HMMT Feb 2026; and five seeds on AIME 2026. These extra seeds are earlier run directories rather than independent draws: on HMMT Feb 2026 and AIME 2026 they share 200–400 of their 480 warm-up traces with seed 0, and the three AIME 2025 seeds share 840 of 848 traces. Averaging per (model, question) unit (Appendix C) keeps them from counting as replicates. Its other three contests have one new seed. gpt-oss-20b and Qwen3-30B use three new seeds on every contest, implemented as disjoint blocks of 64 samples, including all 33 HMMT Feb 2026 questions.

Benchmarks and splits. We use nine competition-mathematics contests from MathArena (Balunović et al., 2025). The development split contains AIME 2024, HMMT Feb 2025, BRUMO 2025, SMT 2025, and CMIMC 2025 (183 questions). The test split contains AIME 2025, HMMT Nov 2025, HMMT Feb 2026, and AIME 2026 (123 questions); the last three postdate every evaluated model. Pooled over models and seeds there are 1,281 development and 1,181 test pools, of which 408 and 381 are contested. Mathematical answers are graded by symbolic equivalence. GPQA Diamond is analyzed separately in Appendix N.

Evidence sampling. Programs are sampled for every question. The candidate-conditioned controls (re-solves, programs and verification) are sampled for questions that are contested in at least one seed. Evidence is sampled once per question and therefore shared across that question's sampling seeds. In deployment, only contested questions need auxiliary evidence; token-cost comparisons use this adaptive accounting.

Program execution. Programs run in a sandbox with a 60 s limit and access only to the Python standard library, `sympy`, and `numpy`. Invalid or non-answering programs abstain. Exact prompts are given in Appendix V; sampler, sandbox, grading and token details in Appendix W.

B Auxiliary Evidence Sources and Override Rules

All sources are sampled once per question with the pool's sampling settings (Appendix A); the exact prompts are in Appendix V.

Candidate-conditioned re-solving. The base model solves the problem again while shown the top-5 candidates in random order, with the instruction that at most one of them is correct (the "hinted" prompt of Appendix V). We sample $n = 8$ re-solves; their answers form a repeated-answer source like any other.

Candidate verification. The base model judges each top-5 candidate correct or incorrect, with $n = 4$ independent verification samples per candidate. Each "correct" verdict counts as one vote for its candidate and "incorrect" verdicts count for nothing; s is the candidate with the most votes (a tie between candidates abstains) and σ is its share of all "correct" verdicts. Unanimity therefore means that every "correct" verdict went to the same candidate. The minimum $m = 4$ counts answered verdicts over all candidates (8 to 20 per question), so a single "correct" verdict among rejections can trigger an override: of the 12 unanimous overrides, 6 rest on one "correct" verdict, 3 on two, 2 on three and 1 on four, and in every one all answered verdicts on the vote's candidate were "incorrect". A stricter rule, which requires the vote to be rejected by all of its $k \geq 3$ verdicts and exactly one other candidate to be accepted by all of its, fires on about 1% of contested pools and gains $+0.1$ points on each split (Appendix P).

Own programs. The base model writes a Python program intended to compute the answer. A sandboxed interpreter executes the program, and the value on its last `ANSWER:` line is the answer. We sample $n = 8$ programs.

Candidate-conditioned programs. The base model writes $n = 8$ programs while shown the candidates. This tests whether the usefulness of execution depends on conditioning the program generator on the existing candidate set.

Tool-integrated traces. For gpt-oss-20b, whose chat template includes a Python tool, we also sample eight traces per contested question in which the model runs code while it reasons; the answer is the trace's final boxed answer (Appendix W). This source is available for one model only, and the tables label it as such. We tried to extend it: offered the same Python tool through its native tool-call format, Qwen3-30B-Thinking made no tool call in its first 32 samples (two shards), and prompted in the ToRA format (Gou et al., 2024) neither Qwen3-30B nor R1-8B wrote executable code blocks while reasoning; R1-8B's chat template has no tool interface. Tool-integrated traces as a base solver therefore remain a limitation of this study.

Cross-model sources. We evaluate both programs and 64-trace reasoning pools from another model family. We also consider an equal-budget source consisting of eight traces drawn from the other two models, four from each.

Confidence and intervention rules. Every source above except the tournament is used through the override rule of Section 2.2: with plurality answer s and share σ among the answering samples, the rule outputs s when at least m samples answer, $\sigma \geq \theta$ and $s \not\equiv \hat{a}$. We use $\theta \in \{0.5, 0.75, 1\}$ and $m = 6$ for eight-sample sources ($m = 32$ for a 64-trace pool). An override is *open* when its answer

may lie outside the pool's top-5 candidates and *candidate-restricted* otherwise. The unanimous program override (Section 2.2) is the open override with own programs at $\theta = 1$, $m = 6$; the pre-registered primary rule is its candidate-restricted version.

Combination rules and jury. Table 11 also includes rules that combine the pool and the programs symmetrically rather than overriding. *Programs alone* replaces the vote by the plurality answer of the programs that execute to an answer (exit without error or timeout and print a computed value, not a bare literal); a single answering program suffices, and a tie for the top answer or no answering program keeps the vote. There is no minimum of six answering programs, which applies only to the override rules. The other sources used alone follow the same convention: verification's candidate with the most "correct" verdicts (at least one; a tie keeps the vote), and the plurality of the answering re-solves, cross-model traces, programs or pool traces. CPMaj, CPMax and CPAgr are the majority, maximum-confidence and agreement rules with which Saparkhan et al. (2026) combine chain-of-thought and program votes, applied to the pool and the eight programs. With *counts* each sample is one vote, so eight programs rarely outvote 64 traces; with *shares* each modality's votes are normalized to shares before they are combined. The *jury* (Liu, 2026) combines the pools of all three models of the same seed, either by a majority over the models' votes or by summing their answer shares.

C Protocol History and Statistical Analysis

Protocol history. Before any rescue result, we fixed the evidence sources, rules, thresholds, development/test split, and decision criteria in a written protocol. The candidate-restricted unanimous program rule was the primary rule. The open unanimous variant was also included in the protocol's rule list.

The open rule was identified as the working rule on the R1-8B pools, whose test contests had already been used in the tournament experiments, and was then fixed before any gpt-oss-20b or Qwen3-30B pool was scored. Those two models are therefore held out for that rule, although two of their three improved test model–question pairs occur on problems also rescued for R1-8B.

The strict program prompt in Appendix V replaced an earlier lenient prompt after the lenient version produced literal-printing programs. This change was made before any gpt-oss-20b or Qwen3-30B pool was scored.

Three later decisions were made after test results had been seen and are treated as exploratory: the $\theta \in \{0.5, 0.75\}$ variants with six answering programs, the question-level statistical summaries, and the rule-selection analyses in Appendix L. The protocol's first implementation of the plurality rules allowed as few as two answering programs; its behavior is reported in Appendix K.

After every rule and threshold had been fixed, gpt-oss-20b was run on a tenth, question-disjoint contest (ArXivMath 2026; Appendix T), and all three models on five further contests (Appendix R).

Statistical unit. A question's auxiliary evidence is shared across its sampling seeds, so seeds are not independent replicates. Our primary unit is therefore a (model, question) pair, with accuracy changes averaged across its seeds. We report seed-averaged whole-benchmark gain with a 95% bootstrap interval that resamples questions, keeping all models and seeds of the same question together. We additionally report the numbers of model–question units improved and worsened, exact sign tests over model–question units and over distinct questions, and pool-level rescue/regression counts as descriptive totals.

The rule comparison of Table 11 contains many comparisons and its intervals are not corrected for multiplicity (Holm-adjusted tests are in Appendix R). The only comparison fixed in advance is the unanimous execution rule against majority voting. Appendix U repeats the program analysis with an independent block of programs per seed.

Pairwise tournament. R1-8B's test pools were first used for a pairwise tournament in which the base model judged condensed evidence for its top answer clusters in a single-elimination bracket. Across seven judge configurations it netted between 0 and $+4$ of 90 mathematics questions, and inside the adaptive pipeline 0, 0 and -1 questions over three runs; we do not analyze it further.

D PRECISION UNDER DISAGREEMENT BY SPLIT

Table 7 repeats the main precision-under-disagreement analysis separately for development and test. The ordering of evidence sources is the same on both splits, while ρ_{dis} is higher on development, where the vote is more often correct.

Table 7: Table 17 separately for development and test contests (contested pools, three models pooled). Disagreements: overrides, with (model, question) pairs in parentheses; π_{dis}, ρ_{dis}, $A_{\text{dis}} = \pi_{\text{dis}} - \rho_{\text{dis}}$ in points, and net = rescues − regressions.

		development					test				
source	θ	disagreements	π_{dis}	ρ_{dis}	A_{dis}	net	disagreements	π_{dis}	ρ_{dis}	A_{dis}	net
Same-model evidence											
Candidate-conditioned re-solve	0.5	74 (40)	26%	39%	−13	−10	68 (24)	29%	10%	+19	+13
	0.75	12 (7)	25%	50%	−25	−3	15 (6)	67%	0%	+67	+10
	1	0	—	—	—	0	1 (1)	0%	0%	+0	+0
Candidate verification	0.5	39 (24)	15%	62%	−47	−18	48 (23)	46%	10%	+36	+17
	0.75	13 (8)	23%	54%	−31	−4	19 (9)	37%	16%	+21	+4
	1	5 (4)	20%	60%	−40	−2	7 (5)	14%	43%	−29	−2
Executed computation											
Own executed programs	0.5	52 (27)	71%	12%	+59	+31	54 (17)	61%	15%	+46	+25
	0.75	32 (17)	75%	6%	+69	+22	37 (11)	81%	0%	+81	+30
	1	21 (10)	86%	5%	+81	+17	25 (8)	96%	0%	+96	+24
Cross-model evidence											
Other-model executed programs	0.5	125 (43)	66%	19%	+47	+58	131 (36)	64%	12%	+52	+68
	0.75	68 (24)	68%	22%	+46	+31	94 (23)	85%	6%	+79	+74
	1	32 (17)	88%	0%	+88	+28	51 (13)	98%	0%	+98	+50
Other-model reasoning pool	0.5	95 (53)	64%	25%	+39	+37	96 (40)	59%	21%	+38	+37
	0.75	40 (20)	75%	20%	+55	+22	37 (18)	70%	5%	+65	+24
	1	10 (5)	70%	30%	+40	+4	2 (1)	100%	0%	+100	+2
Other-model reasoning (8 traces)	0.5	41 (38)	59%	27%	+32	+13	46 (31)	52%	17%	+35	+16
	0.75	16 (15)	88%	12%	+76	+12	19 (13)	74%	0%	+74	+14
	1	5 (5)	100%	0%	+100	+5	1 (1)	0%	0%	+0	+0

On development contests, the same-model sources lose under most thresholds: verification nets -18, -4, and -2 at the three thresholds, while candidate-conditioned re-solving nets -10 and -3. Own programs instead net $+31$, $+22$, and $+17$.

On the harder test contests, ρ_{dis} falls for most sources and loose same-model rules can obtain small positive net gains; verification at $\theta = 0.5$, for example, nets $+17$. The confidence trend nevertheless remains unfavorable for verification: 46%, 37%, and 14% as the threshold tightens. Own programs move in the opposite direction: 61%, 81%, and 96%. The same-model gains on test therefore come mostly from ρ_{dis} (though for re-solving at $\theta = 0.75$ π_{dis} also rises): at $\theta = 0.5$ verification's disagreement advantage moves from -46 [-77, -11] on development to $+35$ [$+0$, $+65$] on test. Appendix M examines threshold selection from development data under an explicit regression-risk criterion.

E DISAGREEMENT ADVANTAGE ACROSS THRESHOLDS

Figure 2 (seven thresholds) and Table 8 (four) extend Table 17 and give 95% intervals for π_{dis}, ρ_{dis} and their difference A_{dis} from one paired bootstrap that resamples distinct contest questions, all models and seeds of a question together (the question counts in parentheses count a question once across models). Coverage is the share of the 789 contested pools (408 development, 381 test) that the rule overrides. For programs, own or another model's, the interval of A_{dis} excludes zero at every threshold on both splits, and π_{dis} rises as coverage falls. For the same-model sources the pooled interval includes zero at every threshold. By split, verification at $\theta = 0.5$ is harmful on development (-46 [-77, -11]) and borderline helpful on test ($+35$ [$+0$, $+65$]); candidate-conditioned re-solving at $\theta = 0.75$ (Table 7) has a positive interval on test ($+67$ [$+10$, $+96$], 15 overrides on six questions) and a negative estimate on development (-25 [-86, $+50$]). In each case the same-model π_{dis} stays near or below one half and the sign of A_{dis} follows ρ_{dis}, the vote's precision on the overridden pools, which is what distinguishes the two contest sets (Section 4.2).

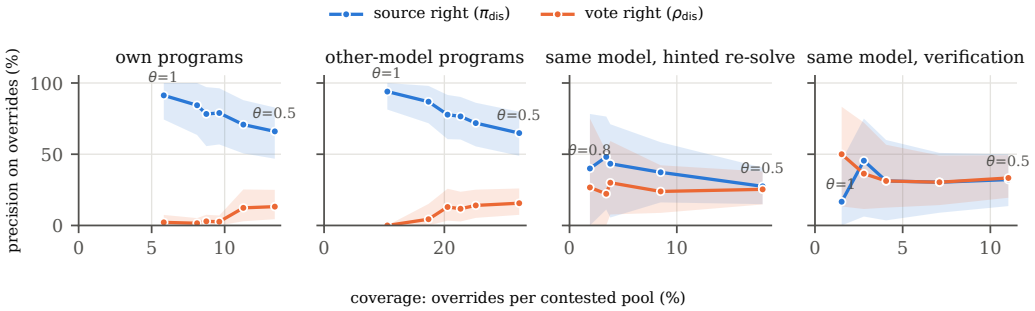


Figure 2: Precision under disagreement against coverage, contested pools of both splits, three models pooled. One point per threshold $\theta \in \{0.5, 0.6, 0.7, 0.75, 0.8, 0.9, 1\}$ (thresholds with identical overrides merged; points with fewer than five overrides omitted). Blue: the source is right on its overrides (π_{dis}); orange: the overruled vote is right on the same overrides (ρ_{dis}); the vertical gap is A_{dis}. Bands: 95% bootstrap intervals over questions.

Table 8: Disagreement advantage by threshold (contested pools, three models pooled; four of the seven thresholds of Figure 2, full grid in the supplementary file `adv_curve.md`). Overrides (distinct questions, counted once across models); coverage = overrides / contested pools; 95% intervals from a paired bootstrap over distinct questions.

				both splits			development	test
source	θ	disagreements	coverage	π_{dis}	ρ_{dis}	A_{dis} [95% CI]	A_{dis} [95% CI]	A_{dis} [95% CI]
Same-model evidence								
Candidate-conditioned re-solve	0.5	142 (57)	18.0%	27%	25%	+2 [−18, +21]	−14 [−40, +13]	+19 [−8, +46]
	0.7	30 (15)	3.8%	43%	30%	+13 [−47, +58]	−36 [−92, +33]	+56 [−11, +90]
	0.8	15 (6)	1.9%	40%	27%	+13 [−75, +75]	−80 [−100, +0]	+60 [+0, +100]
	1	1 (1)	0.1%	0%	0%	+0 [+0, +0]	—	+0 [+0, +0]
Candidate verification	0.5	87 (42)	11.0%	32%	33%	−1 [−32, +26]	−46 [−77, −11]	+35 [+0, +65]
	0.7	32 (16)	4.1%	31%	31%	+0 [−46, +43]	−31 [−82, +27]	+21 [−43, +67]
	0.8	22 (12)	2.8%	45%	36%	+9 [−60, +60]	−22 [−90, +60]	+31 [−67, +85]
	1	12 (9)	1.5%	17%	50%	−33 [−77, +20]	−40 [−100, +50]	−29 [−86, +40]
Executed computation								
Own executed programs	0.5	106 (35)	13.4%	66%	13%	+53 [+26, +76]	+60 [+25, +90]	+46 [+2, +79]
	0.7	76 (25)	9.6%	79%	3%	+76 [+52, +96]	+69 [+35, +97]	+82 [+51, +100]
	0.8	64 (20)	8.1%	84%	2%	+83 [+61, +100]	+85 [+54, +100]	+81 [+50, +100]
	1	46 (16)	5.8%	91%	2%	+89 [+67, +100]	+81 [+41, +100]	+96 [+81, +100]
Candidate-conditioned programs	0.5	140 (48)	17.7%	59%	10%	+49 [+30, +67]	+42 [+12, +70]	+54 [+28, +77]
	0.7	81 (27)	10.3%	78%	10%	+68 [+40, +90]	+58 [+18, +94]	+77 [+38, +100]
	0.8	58 (18)	7.4%	84%	3%	+81 [+56, +98]	+58 [+17, +100]	+97 [+88, +100]
	1	39 (14)	4.9%	79%	5%	+74 [+42, +96]	+40 [−7, +87]	+96 [+81, +100]
Tool-integrated traces (gpt-oss-20b only)	0.5	35 (13)	4.4%	91%	0%	+91 [+73, +100]	+100 [+100, +100]	+83 [+50, +100]
	0.7	26 (9)	3.3%	100%	0%	+100 [+100, +100]	+100 [+100, +100]	+100 [+100, +100]
	0.8	21 (7)	2.7%	100%	0%	+100 [+100, +100]	+100 [+100, +100]	+100 [+100, +100]
	1	9 (3)	1.1%	100%	0%	+100 [+100, +100]	+100 [+100, +100]	+100 [+100, +100]
Cross-model evidence								
Other-model executed programs	0.5	256 (55)	32.4%	65%	16%	+49 [+25, +70]	+46 [+13, +78]	+52 [+19, +78]
	0.7	179 (38)	22.7%	77%	12%	+65 [+39, +86]	+47 [+10, +81]	+80 [+45, +100]
	0.8	137 (29)	17.4%	87%	4%	+82 [+58, +98]	+70 [+21, +100]	+90 [+71, +100]
	1	83 (22)	10.5%	94%	0%	+94 [+81, +100]	+88 [+62, +100]	+98 [+91, +100]
Other-model reasoning pool	0.5	191 (61)	24.2%	62%	23%	+39 [+24, +54]	+39 [+18, +60]	+39 [+19, +60]
	0.7	91 (39)	11.5%	71%	14%	+57 [+33, +78]	+50 [+14, +81]	+64 [+38, +89]
	0.8	57 (27)	7.2%	68%	11%	+58 [+31, +82]	+55 [+13, +88]	+62 [+31, +95]
	1	12 (6)	1.5%	75%	25%	+50 [−38, +100]	+40 [−64, +100]	+100 [+100, +100]
Other-model reasoning (8 traces)	0.5	87 (57)	11.0%	55%	22%	+33 [+15, +51]	+32 [+0, +62]	+35 [+14, +57]
	0.7	36 (29)	4.6%	81%	6%	+75 [+55, +91]	+76 [+44, +100]	+74 [+47, +95]
	0.8	21 (18)	2.7%	81%	10%	+71 [+43, +95]	+75 [+36, +100]	+67 [+30, +100]
	1	6 (6)	0.8%	83%	0%	+83 [+50, +100]	+100 [+100, +100]	+0 [+0, +0]

F HEADROOM BY NUMBER OF CANDIDATES

Table 9 repeats the headroom analysis of Table 2 with the K highest-ranked answer clusters as candidates instead of five ("pool": any trace of the pool). On the test contests a wrong contested vote has its correct answer among the top 2, 3, 5 and 10 candidates in 22%, 32%, 38% and 44% of pools, and anywhere in the pool in 50%; per (model, question) unit the shares are 22%, 34%, 40% [26, 56], 44% and 49% [34, 66]. The choice $K = 5$ therefore does not drive the answer-present finding: on test even a perfect selector over the whole pool could correct about half of the wrong contested votes, and doubling K moves the pool-level share only from 38% to 44%. On the development contests the pool contains more of the answers (78%), and the gap between $K = 5$ and the whole pool is larger.

Table 9: Headroom by the number of candidates K (contested-pool definitions as in Table 2; candidates ranked as there). Oracle: share of all pools whose top-K candidates contain the correct answer. Answer present: share of wrong contested votes whose correct answer is among the top-K candidates (pools); the last row of each split averages the share over (model, question) units.

			oracle accuracy, top-K (%)						answer present, top-K (%)				
model	split	vote	2	3	5	10	pool	wrong	2	3	5	10	pool
R1-8B	dev	78.7	84.2	85.2	88.0	92.3	95.1	33	24	30	45	70	85
gpt-oss-20b	dev	87.1	92.5	93.4	94.0	94.7	96.4	66	42	50	55	61	74
Qwen3-30B	dev	87.2	90.9	91.8	92.7	93.8	95.1	52	38	46	56	67	79
all models	dev	85.9	90.6	91.6	92.6	94.0	95.6	151	37	44	53	65	78
dev, per (model, question)									38	45	54	69	82
R1-8B	test	70.9	76.3	78.8	81.5	83.1	84.9	101	22	32	43	50	56
gpt-oss-20b	test	84.6	87.8	89.4	89.4	90.0	91.1	50	22	34	34	38	44
Qwen3-30B	test	85.9	88.3	89.2	89.4	90.0	91.1	37	22	30	32	35	41
all models	test	79.8	83.7	85.4	86.5	87.4	88.7	188	22	32	38	44	50
test, per (model, question)									22	34	40	44	49

G ERROR DEPENDENCE

Table 10 reports how strongly each auxiliary source's errors depend on the majority vote's errors. No source is close to independent. By ϕ, program sources tend to be less dependent than same-model sources, but the ranges overlap, and relative risk is unstable when a source is rarely wrong while the vote is right. The table covers every complete pool, settled ones included, without a confidence threshold, whereas π_{dis} is measured on thresholded overrides of contested pools. The error indicators also count two wrong answers as dependent even when the answers differ, although a source that repeats the vote's wrong answer cannot cause an override.

H REGRESSIONS, COST-WEIGHTED VALUE AND THE GAIN SPLIT

Table 4 (Section 4.3) pools every split on which each rule was run and gives the (model, question) pairs it improves and worsens, the share of its changes that break a right answer, and the value improved minus λ times worsened, where λ is the cost of breaking a right answer relative to fixing a wrong one. On development and test, the ranking depends on λ, with crossovers that rest on a few pairs: the jury by summed shares leads at $\lambda = 1$, the jury by majority of models up to $\lambda = 3$, and programs alone and the $\theta = 0.5$ gate trail the $\theta = 0.75$ gate at every $\lambda \geq 1$. Across 32 rules in 13 source families (Table 23; Spearman, bootstrap over families), development worsened pairs rank test worsened pairs (0.88 [0.79, 0.94]; 0.54 [0.27, 0.77] given coverage), the worsened share of changes carries over (0.67 [0.40, 0.87]), and development gain does not detectably predict it (0.09 [−0.32, +0.57]). On the confirmatory contests, rules with development $\rho_{\text{dis}} \leq 9\%$ worsen no pair, CoT–PoT voting and programs alone 10 and 12 (Table 19).

Figure 3 shows the gain split of Eq. (3) along each source's gate path (values with intervals in Table 21). Under the canonical convention the own programs' accuracy gap is close to zero at eight

Table 10: Error dependence between each source's plurality answer and the vote, on every complete pool (development and test). Relative risk $= P(\text{source wrong} \mid \text{vote wrong})/P(\text{source wrong} \mid \text{vote right})$; ϕ = correlation of the two error indicators. Independent errors would give a relative risk near 1 and ϕ near 0. Programs are sampled once per question, so a program source contributes the same answers under every model's pools; its rows differ across models only through the vote it is compared with.

source	n	error if vote wrong	error if vote correct	rel. risk	ϕ
R1-8B pools					
Own executed programs	535	0.74	0.12	6.0	+0.58
gpt-oss-20b reasoning pool	472	0.46	0.02	18.6	+0.56
Qwen3-30B reasoning pool	474	0.48	0.01	58.2	+0.62
Other-model reasoning (8 traces)	447	0.45	0.01	53.6	+0.60
gpt-oss-20b executed programs	572	0.32	0.01	47.2	+0.49
Qwen3-30B executed programs	550	0.52	0.09	5.5	+0.45
Candidate-conditioned programs	205	0.59	0.23	2.5	+0.35
Candidate-conditioned re-solve	221	0.83	0.05	16.4	+0.77
Candidate verification	197	0.78	0.16	4.9	+0.62
gpt-oss-20b pools					
Own executed programs	852	0.52	0.03	17.0	+0.55
Qwen3-30B reasoning pool	913	0.67	0.04	15.6	+0.64
R1-8B reasoning pool	565	0.85	0.13	6.3	+0.58
Other-model reasoning (8 traces)	844	0.71	0.04	17.0	+0.66
Qwen3-30B executed programs	804	0.59	0.12	5.0	+0.40
R1-8B executed programs	786	0.66	0.19	3.4	+0.33
Candidate-conditioned programs	360	0.51	0.09	5.7	+0.46
Candidate-conditioned re-solve	342	0.87	0.17	5.1	+0.64
Candidate verification	317	0.86	0.10	8.4	+0.71
Tool-integrated traces	366	0.29	0.00	83.6	+0.48
Qwen3-30B pools					
Own executed programs	804	0.74	0.11	7.0	+0.51
gpt-oss-20b reasoning pool	907	0.69	0.05	14.1	+0.63
R1-8B reasoning pool	565	0.93	0.13	7.0	+0.62
Other-model reasoning (8 traces)	816	0.80	0.07	11.1	+0.61
gpt-oss-20b executed programs	852	0.49	0.03	15.4	+0.53
R1-8B executed programs	786	0.74	0.18	4.1	+0.40
Candidate-conditioned programs	168	0.71	0.25	2.9	+0.45
Candidate-conditioned re-solve	213	0.86	0.06	13.4	+0.80
Candidate verification	136	0.67	0.09	7.5	+0.56

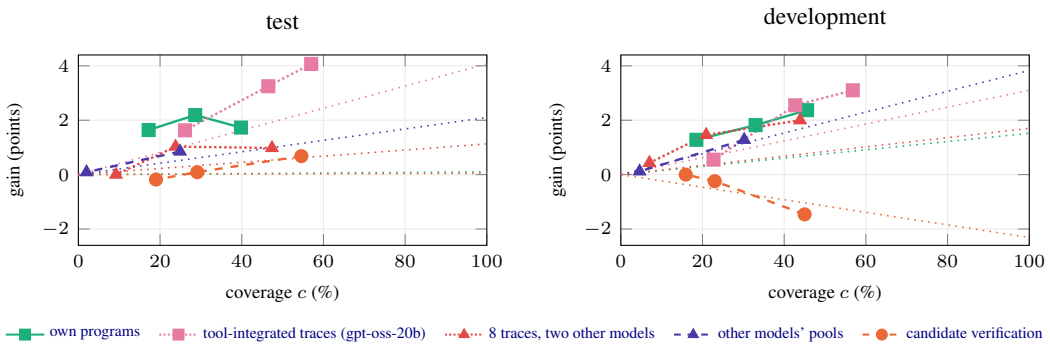


Figure 3: Gain against coverage along each source's gate path ($\theta = 0.5, 0.75, 1$ from right to left; unit-weighted, contested pools of the three models; tool-integrated traces: gpt-oss-20b). The dotted line of each source is what a random gate of the same coverage would earn, $c\,(q - p)$; a point's height above its line is the selectivity term (values with intervals: Table 21).

programs and their gain is selectivity; tool-integrated traces and other models' pools gain mainly through the gap; the same-model sources show neither term. The split moves with the convention for the source alone and with the program budget (Appendix P); $\pi_{\text{dis}} - \rho_{\text{dis}}$ does not.

I Full Program and Combination Results

Table 11 reports the full comparison of program-based rules, same-model controls, resampling, and cross-model evidence; the rules are defined in Appendix B, and Section 4.4 summarizes the main results.

Selective program overrides. Eight additional traces from the original model change almost nothing: resampling gains only $+0.3$ points on test. The eight programs alone are also not a stronger general solver than the vote: 13 test units improve while 17 worsen.

The useful behavior appears in their disagreements with the vote. At every agreement threshold, overriding only when the program answer differs from the majority gives a positive net balance. On test, $\theta = 0.5$ gains $+1.7$ points with 3 worsened units, $\theta = 0.75$ gains $+2.2$ with 0, and unanimity gains $+1.6$ with 0. On development the corresponding gains are $+2.4$, $+1.8$, and $+1.3$, with 4, 2, and 1 worsened units.

Candidate restriction and open answers. Restricting program overrides to answers already in the candidate set removes most of the gain: the candidate-restricted rule gives $+0.4$ under unanimity and -0.3 at $\theta = 0.5$.

Across both splits, of the unanimous own-program rule's 42 rescues, 23 output an answer absent from the pool's top-five candidates, and these open overrides are 88% precise. By model, this accounts for 17 of 19 R1-8B rescues, 5 of 14 gpt-oss-20b rescues, and 1 of 9 Qwen3-30B rescues.

Published combination rules and cross-model jury. Share-normalized CoT–PoT majority voting over the same programs gains $+2.6$ points on test with 6 worsened units and $+1.6$ on development. Maximum-confidence and agreement variants gain $+1.6$ and $+1.7$ on test with 8 and 9 worsened units. With raw counts, eight programs rarely outvote the 64 original traces, giving $+0.9$.

A three-model jury gains $+3.0$ on test and $+3.8$ on development at roughly three times the sampling. Most of the gain occurs for R1-8B, where the two other models are stronger: 13 test units improve and 0 worsen.

Equal-compute comparisons. Spending the program budget on eight additional traces from the original model's next sample block changes 3 test pools for the better and none for the worse, and

Table 11: Combining the vote with other evidence on contested pools. A question's evidence is shared by its seeds, so the unit of analysis is a (model, question) with its per-seed changes averaged. Per model: test units that improve / worsen. Test: seed-averaged whole-benchmark gain over the vote (points) with a 95% bootstrap CI clustered by question across models, and pool-level rescues / regressions. Development: seed-averaged gain (units improved / worsened). Override rules need ≥ 6 answering samples, as in Table 3; $\theta = 1$ is unanimity; open: the output may lie outside the pool's top-5 candidates. CPMaj/CPMax/CPAgr: Saparkhan et al. (2026) over the pool and the programs (counts, or per-modality shares); jury: cross-model consensus (Liu, 2026) with the other two models' pools of the same seed. §: the pre-registered primary rule. Tool-integrated traces exist for gpt-oss-20b only (its chat template has a Python tool); the pooled columns average its gain over all three models' units.

		test units +/−			test		development
source	rule	R1-8B	gpt-oss-20b	Qwen3-30B	Δ [95% CI]	pools +/−	Δ (units +/−)
Additional same-model sampling							
8 additional traces	re-vote	0/0	2/0	1/0	+0.3 [+0.0, +0.6]	+3/−0	+0.1 (4/3)
Same-model correction							
Candidate-conditioned re-solve	override $\theta = 0.5$	3/0	1/2	2/1	+0.5 [−0.7, +1.8]	+20/−7	−0.1 (11/15)
	override $\theta = 0.75$	2/0	0/0	1/0	+0.5 [+0.0, +1.2]	+10/−0	−0.2 (1/3)
Candidate verification	unanimous override	0/1	1/1	0/0	−0.2 [−0.6, +0.2]	+1/−3	+0.0 (1/2)
Executed computation: own programs							
Own executed programs	programs alone (plurality)	5/8	6/3	2/6	−0.9 [−3.9, +1.9]	+40/−40	+1.1 (24/16)
	CPMaj, counts	2/0	4/0	1/0	+0.9 [+0.2, +1.8]	+11/−0	+0.5 (8/2)
	CPMaj, shares	7/0	7/3	5/3	+2.6 [−0.1, +5.6]	+53/−14	+1.6 (24/13)
	CPMax, shares	6/0	7/3	3/5	+1.6 [−1.1, +4.7]	+48/−20	+1.6 (21/13)
	CPAgr, shares	7/3	6/3	5/3	+1.7 [−0.9, +4.4]	+50/−17	+1.4 (25/15)
	override $\theta = 0.5$, top-5 only	1/1	1/0	0/2	−0.3 [−1.4, +0.6]	+5/−8	+1.6 (13/2)
	override $\theta = 1$, top-5 only§	1/0	1/0	0/0	+0.4 [+0.0, +1.0]	+5/−0	+0.9 (7/0)
	override $\theta = 0.5$, open	4/1	4/0	2/2	+1.7 [−0.3, +4.0]	+33/−8	+2.4 (18/4)
	override $\theta = 0.75$, open	4/0	3/0	2/0	+2.2 [+0.5, +4.2]	+30/−0	+1.8 (13/2)
	override $\theta = 1$, open	4/0	2/0	1/0	+1.6 [+0.3, +3.3]	+24/−0	+1.3 (9/1)
Executed computation: tool-integrated traces							
Tool-integrated traces (gpt-oss-20b only)	traces alone (plurality)	0/0	9/0	0/0	+1.9 [+0.7, +3.4]	+21/−0	+2.0 (15/1)
	override $\theta = 0.75$, open	0/0	4/0	0/0	+1.1 [+0.3, +2.2]	+12/−0	+0.9 (5/0)
	override $\theta = 1$, open	0/0	2/0	0/0	+0.5 [+0.0, +1.4]	+6/−0	+0.2 (1/0)
Cross-model evidence							
Other-model reasoning (8 traces)	override $\theta = 0.75$, open	6/0	2/0	1/0	+1.0 [+0.3, +1.9]	+14/−0	+1.5 (12/1)
	override $\theta = 1$, open	0/0	0/0	0/0	+0.0 [+0.0, +0.0]	+0/−0	+0.4 (5/0)
Other-model reasoning pools	jury, majority of models	11/0	2/1	1/1	+1.9 [+0.6, +3.3]	+31/−4	+3.0 (22/2)
(3× sampling)	jury, summed shares	13/0	5/2	4/3	+3.0 [+1.1, +5.2]	+46/−9	+3.8 (34/13)

gains $+0.1$ on development (4/3 units). Empirically, these extra draws rarely move an already-contested 64-trace vote.

Drawing the same eight traces from the two other models, four from each, is more useful. Overriding at $\theta = 0.75$ gains $+1.0$ on test (9/0 units) and $+1.5$ on development (12/1), primarily for R1-8B (6/0 test units). Under unanimity the corresponding gains are $+0.0$ on test and $+0.4$ on development. At the same eight-sample budget and without requiring a second model, own programs gain $+2.2$ on test at $\theta = 0.75$.

Cost. Eight programs cost 135k output tokens per contested question for gpt-oss-20b and 174k for Qwen3-30B, compared with 1.59M and 2.20M for their 64-trace pools. Relative to the full adaptive protocol—16 traces for settled questions, 64 for contested questions, and programs only for contested questions—the rescue adds 7.6–7.8% of tokens for gpt-oss-20b and 5.3–5.7% for Qwen3-30B across the three seeds, and 4.8% for R1-8B on the three contests whose pools were sampled with token logs.

J Per-Model and Per-Contest Results

Table 12 breaks the unanimous open execution rule down by model and contest, including whole-contest majority accuracy, top-five oracle accuracy, and pool-level rescue/regression counts. The intervals resample questions within one model and are descriptive. The pooled gains here weight every pool equally, whereas Tables 11 and 26 average each question over its seeds; the two differ where questions have different numbers of seeds, as for R1-8B's test contests ($+3.8$ points here against $+3.1$ in Table 26).

The pooled test oracles of gpt-oss-20b and Qwen3-30B in Table 2 coincide (top-5 oracle 330 and pool oracle 336 of 369 pools for both models), although their per-contest values differ.

K Sensitivity Analyses

All values below pool the three models unless otherwise stated.

Minimum answering programs. Under unanimity, $m = 4$ gives the same results as $m = 6$ (test $+1.6$, development $+1.3$), while $m = 8$ gives $+1.6$ and $+0.9$.

Candidate restriction. Allowing only candidate answers reduces unanimity to $+0.4$ on test and $+0.9$ on development, and reduces the $\theta = 0.75$ rule to $+0.4$ and $+1.2$.

Original plurality-rule implementation. The protocol's first implementation allowed as few as two answering programs to trigger a plurality override. With that setting, the candidate rule at $\theta = 0.5$ gains $+1.5$ on development (17/5 units) and -1.1 on test (3/7). The open rule at $\theta = 0.75$ gains $+1.9$ on test (9/1). Requiring six answering programs, as in Table 11, removes most of these regressions.

Applying rules to settled pools. Applying the rules to every pool changes their risk. The unanimous rule gains $+1.9$ on test (7/0 units) and $+1.3$ on development (9/1). The $\theta = 0.75$ override gives $+2.2$ (9/1) and $+1.5$ (14/4); the $\theta = 0.5$ override gives $+0.5$ (10/8) and $+1.5$ (20/10). Share-normalized CPMaj gives $+2.2$ (19/9) and $+1.7$ (25/13). The contested gate is therefore mainly a cost saving for unanimity and a safety device for the looser rules.

Warm-up and settling threshold. The adaptive protocol (16 warm-up traces, settle at a top share $\geq \gamma$ with $\gamma = 0.8$) saves cost and does not create the gains. On the gpt-oss-20b and Qwen3-30B pools of all nine contests and three seeds, where all 64 traces exist, its vote is correct on exactly the same pools as the 64-trace vote (no change of correctness in 1,239 settled pools) at 76–77% (gpt-oss-20b) and 54–55% (Qwen3-30B) of the trace tokens, and the gate sensitivity above shows that the unanimous rule rescues the same units when settled pools are opened. Varying the threshold γ over the same 1,836 pools, early stopping loses nothing for every $\gamma \geq 0.7$ and changes 3 and 13

Table 12: The unanimous program override per model and contest (open, $\theta = 1$, m=6, contested pools). Pools = questions $\times$ seeds; vote and +override are whole-contest accuracies; oracle = a top-5 candidate is correct; rescues and regressions are pool counts; the pooled interval is clustered by question within the model. † held after every model's release.

contest	pools	contested	vote	+override	oracle top-5	rescues / regr.
DeepSeek-R1-0528-Qwen3-8B (1–6 seeds)						
AIME 2024	30	4	90.0	90.0	93.3	+0 / −0
HMMT Feb 2025	30	14	73.3	73.3	80.0	+0 / −0
BRUMO 2025	30	9	96.7	96.7	96.7	+0 / −0
SMT 2025	53	16	75.5	75.5	88.7	+0 / −0
CMIMC 2025	40	23	65.0	70.0	82.5	+2 / −0
dev pooled	183	66	78.7	**79.8**	88.0	+2 / −0
gain +1.1 pp, 95% CI [+0.0, +2.7]						
AIME 2025	83	21	85.5	89.2	91.6	+3 / −0
HMMT Nov 2025†	30	13	83.3	83.3	90.0	+0 / −0
HMMT Feb 2026†	180	85	51.7	53.9	70.6	+4 / −0
AIME 2026†	150	32	83.3	90.0	87.3	+10 / −0
test pooled	443	151	70.9	**74.7**	81.5	+17 / −0
gain +3.8 pp, 95% CI [+0.7, +8.1]						
gpt-oss-20b (seeds 0–2)						
AIME 2024	90	21	93.3	93.3	94.4	+0 / −0
HMMT Feb 2025	90	41	82.2	82.2	92.2	+0 / −0
BRUMO 2025	90	26	90.0	91.1	100.0	+1 / −0
SMT 2025	159	59	88.1	88.1	90.6	+0 / −0
CMIMC 2025	120	73	82.5	87.5	95.0	+7 / −1
dev pooled	549	220	87.1	**88.3**	94.0	+8 / −1
gain +1.3 pp, 95% CI [−0.2, +3.1]						
AIME 2025	90	37	93.3	96.7	93.3	+3 / −0
HMMT Nov 2025†	90	37	82.2	85.6	91.1	+3 / −0
HMMT Feb 2026†	99	49	68.7	68.7	77.8	+0 / −0
AIME 2026†	90	19	95.6	95.6	96.7	+0 / −0
test pooled	369	142	84.6	**86.2**	89.4	+6 / −0
gain +1.6 pp, 95% CI [+0.0, +4.1]						
Qwen3-30B-A3B-Thinking (seeds 0–2)						
AIME 2024	90	4	93.3	93.3	93.3	+0 / −0
HMMT Feb 2025	90	36	80.0	82.2	92.2	+2 / −0
BRUMO 2025	90	13	96.7	96.7	96.7	+0 / −0
SMT 2025	159	33	84.9	86.8	88.7	+3 / −0
CMIMC 2025	120	36	84.2	86.7	95.0	+3 / −0
dev pooled	549	122	87.2	**88.7**	92.7	+8 / −0
gain +1.5 pp, 95% CI [+0.0, +3.5]						
AIME 2025	90	15	88.9	88.9	95.6	+0 / −0
HMMT Nov 2025†	90	23	86.7	86.7	90.0	+0 / −0
HMMT Feb 2026†	99	37	75.8	75.8	79.8	+0 / −0
AIME 2026†	90	13	93.3	94.4	93.3	+1 / −0
test pooled	369	88	85.9	**86.2**	89.4	+1 / −0
gain +0.3 pp, 95% CI [+0.0, +0.8]						

votes at $\gamma = 0.6$ and 0.5; at $\gamma = 0.8$ the adaptive gate contests 31% of pools (572 of 1,836) and these hold 82% of the 250 wrong votes (pools whose 64-trace top share is below 0.8, ignoring settling at the warm-up, are 35% and hold 85%), and $\gamma = 1$ would catch 95% at about 2.1 times the program cost of the adaptive gate (supplementary files `warmup_ablation.md` and `gamma_sweep.md`).

Candidate-conditioned programs. Programs written while shown the candidates, under the same open unanimous rule, give $+1.5$ on test (6/0 units) and $+0.6$ on development (5/2). Showing candidates neither helps nor is required in these experiments.

Other-model programs. Under the open unanimous rule, every other model's programs are a zero-regression source on every model and split. For R1-8B, gpt-oss-20b programs give $+17/0$ test and $+8/0$ development rescues/regressions, while Qwen3-30B programs give $+19/0$ and $+5/0$. For gpt-oss-20b, Qwen3-30B programs give $+1/0$ and $+7/0$, while R1-8B programs give $+4/0$ and $0/0$. For Qwen3-30B, gpt-oss-20b programs give $+6/0$ and $+8/0$, while R1-8B programs give $+3/0$ and $0/0$. Measured per (model, question) unit, R1-8B's programs improve 2/0 units of gpt-oss-20b ($+1.1$ points [$+0.0$, $+3.0$]) and 2/0 of Qwen3-30B ($+0.8$ [$+0.0$, $+2.2$]) on test, on 2 distinct questions in all (pooled $+0.9$ [$+0.0$, $+2.6$]; sign $p = 0.500$), and never fire on development.

L Selecting Rules from Data

We perform leave-one-model-out selection among the single-model rules in Table 11, excluding the jury and programs-alone conditions. A rule is selected on the other two models' development contests ("dev") or development plus test contests ("dev+test") using the largest seed-averaged gain, the largest lower 2.5% bootstrap bound of that gain, or the largest gain among rules with no worsened model–question unit. The selected rule is then evaluated on the held-out model's test contests. The last row selects using all three models' development contests. This analysis was performed after the rules had already been evaluated on test and is exploratory.

Table 13: Leave-one-model-out rule selection: chosen rule and its result on the held-out model's test contests (units improved / worsened; seed-averaged gain in points).

held-out model	selection data	procedure	chosen rule	held-out units +/−	held-out Δ
gpt-oss-20b	dev	max gain	programs, $\theta = 0.5$, open	4 / 0	+3.25
gpt-oss-20b	dev	max lower bound	programs, $\theta = 0.75$, open	3 / 0	+2.44
gpt-oss-20b	dev	no worsened unit	programs, $\theta = 1$, open	2 / 0	+1.63
gpt-oss-20b	dev+test	max gain	programs, $\theta = 0.75$, open	3 / 0	+2.44
gpt-oss-20b	dev+test	max lower bound	programs, $\theta = 0.75$, open	3 / 0	+2.44
gpt-oss-20b	dev+test	no worsened unit	programs, $\theta = 1$, open	2 / 0	+1.63
Qwen3-30B	dev	max gain	programs, $\theta = 0.5$, open	2 / 2	−0.27
Qwen3-30B	dev	max lower bound	programs, $\theta = 0.75$, open	2 / 0	+1.08
Qwen3-30B	dev	no worsened unit	tool-integrated traces (gpt-oss-20b only), $\theta = 0.5$	0 / 0	+0.00
Qwen3-30B	dev+test	max gain	CPMaj, shares	5 / 3	+0.54
Qwen3-30B	dev+test	max lower bound	programs, $\theta = 0.5$, open	2 / 2	−0.27
Qwen3-30B	dev+test	no worsened unit	tool-integrated traces (gpt-oss-20b only), $\theta = 0.5$	0 / 0	+0.00
R1-8B	dev	max gain	$\theta = 0.5$, candidates, $m = 2$	2 / 1	−0.08
R1-8B	dev	max lower bound	$\theta = 0.5$, candidates, $m = 2$	2 / 1	−0.08
R1-8B	dev	no worsened unit	$\theta = 0.5$, candidates	1 / 1	−0.50
R1-8B	dev+test	max gain	programs, $\theta = 0.5$, open	4 / 1	+2.22
R1-8B	dev+test	max lower bound	tool-integrated traces (gpt-oss-20b only), $\theta = 0.5$	0 / 0	+0.00
R1-8B	dev+test	no worsened unit	tool-integrated traces (gpt-oss-20b only), $\theta = 0.5$	0 / 0	+0.00
none (all, dev)	dev	max gain	programs, $\theta = 0.5$, open	10 / 3	+1.73
none (all, dev)	dev	max lower bound	programs, $\theta = 0.5$, open	10 / 3	+1.73
none (all, dev)	dev	no worsened unit	tool-integrated traces (gpt-oss-20b only), $\theta = 0.5$	5 / 0	+1.37

Selection by development gain does not transfer reliably. When gpt-oss-20b is held out, every procedure picks an open program override that improves its test contests without a worsened unit. When Qwen3-30B is held out, selecting the largest development gain picks the $\theta = 0.5$ override, which improves and worsens two units each on its test contests. When R1-8B is held out, development-only selection picks candidate-restricted plurality rules (twice in the protocol's first implementation with $m = 2$; Appendix K), which lose on its test contests. The procedure "largest gain with no worsened unit" often selects the tool-integrated traces, which exist only for gpt-oss-20b; for another held-out model this choice keeps the vote. Across folds the selected threshold ranges from 0.5 to 1, so no single rule is recovered by data-driven selection; what holds across folds is that whenever an open program override at $\theta \geq 0.75$ is selected, it worsens no held-out test unit.

M RISK-CONTROLLED THRESHOLD SELECTION

We select the override threshold with a fixed-sequence test in the style of Learn-then-Test (Angelopoulos et al., 2021). For a tolerated rate α of worsened (model, question) units and a threshold θ, let n be the number of development units and k the number that the rule worsens. We test H_θ: "the rule's worsened-unit rate exceeds α" with the exact binomial p-value

$$P(X \le k), \qquad X \sim \mathrm{Binomial}(n, \alpha),$$

and reject it when this is at most $\delta = 0.1$. Thresholds are tested in the fixed order $\theta = 1, 0.75, 0.5$, from strict to loose, and testing stops at the first non-rejection. The loosest threshold whose hypothesis was rejected is selected; if H_1 is not rejected, the majority vote is kept. Selection uses either all three models' development contests or, in leave-one-model-out analysis, the other two models. The selected rule is then evaluated on test. This analysis was designed after test results had been seen.

Table 14: Risk-controlled threshold selection for the open and the candidate-restricted program overrides.

answer space	α	selection (development contests)	selected	test units +/−	test Δ
open	0.5%	all models	vote	— (keep the vote)	+0.00
open	0.5%	held out gpt-oss-20b	vote	— (keep the vote)	+0.00
open	0.5%	held out Qwen3-30B	vote	— (keep the vote)	+0.00
open	0.5%	held out R1-8B	vote	— (keep the vote)	+0.00
open	1.0%	all models	programs, $\theta = 0.75$, open	9 / 0 (of 366)	+2.19
open	1.0%	held out gpt-oss-20b	programs, $\theta = 1$, open	2 / 0 (of 123)	+1.63
open	1.0%	held out Qwen3-30B	vote	— (keep the vote)	+0.00
open	1.0%	held out R1-8B	vote	— (keep the vote)	+0.00
open	2.0%	all models	programs, $\theta = 0.5$, open	10 / 3 (of 366)	+1.73
open	2.0%	held out gpt-oss-20b	programs, $\theta = 0.5$, open	4 / 0 (of 123)	+3.25
open	2.0%	held out Qwen3-30B	programs, $\theta = 0.75$, open	2 / 0 (of 123)	+1.08
open	2.0%	held out R1-8B	programs, $\theta = 0.5$, open	4 / 1 (of 120)	+2.22
candidates	0.5%	all models	$\theta = 1$, candidates	2 / 0 (of 366)	+0.38
candidates	0.5%	held out gpt-oss-20b	vote	— (keep the vote)	+0.00
candidates	0.5%	held out Qwen3-30B	vote	— (keep the vote)	+0.00
candidates	0.5%	held out R1-8B	vote	— (keep the vote)	+0.00
candidates	1.0%	all models	$\theta = 0.5$, candidates	2 / 3 (of 366)	−0.35
candidates	1.0%	held out gpt-oss-20b	$\theta = 1$, candidates	1 / 0 (of 123)	+0.81
candidates	1.0%	held out Qwen3-30B	$\theta = 1$, candidates	0 / 0 (of 123)	+0.00
candidates	1.0%	held out R1-8B	$\theta = 0.5$, candidates	1 / 1 (of 120)	−0.50
candidates	2.0%	all models	$\theta = 0.5$, candidates	2 / 3 (of 366)	−0.35
candidates	2.0%	held out gpt-oss-20b	$\theta = 0.5$, candidates	1 / 0 (of 123)	+0.81
candidates	2.0%	held out Qwen3-30B	$\theta = 0.5$, candidates	0 / 2 (of 123)	−1.36
candidates	2.0%	held out R1-8B	$\theta = 0.5$, candidates	1 / 1 (of 120)	−0.50

In every row of Tables 14 and 15, the realized worsened rate of the selected rule on the test contests stays below the tolerance α. At $\alpha = 1\%$ over units, the selection on all three models' development contests picks the $\theta = 0.75$ override (9 improved and no worsened test units, +2.2 points). For the open overrides, $\alpha = 0.5\%$ keeps the vote, and at $\alpha = 2\%$ the selection moves to $\theta = 0.5$ and admits regressions. Leave-one-model-out selections, which have fewer development units, keep the vote more often. The candidate-restricted overrides gain little or lose at every tolerance.

N GPQA DIAMOND

On the R1-8B GPQA Diamond pools (198 questions $\times$ 4 seeds; 353 of 792 pools contested; vote 64.0%), execution does not apply, and the four options make the top-five oracle an artifact (92.7%). Here the same-model picture differs from open-answer mathematics in one important respect: unanimous candidate-conditioned re-solving overrides 13 pools and is correct on all 13, corresponding to $+1.6$ points with interval $[+0.3, +3.5]$.

Unanimous verification nets $+6/-4$, and gpt-oss-20b's pool as a cross-model source, available for the 87 contested pools of its first seed, gives $+1/0$. Looser rules regress more frequently for every source: candidate-conditioned re-solving at plurality ≥ 0.5 gives $+37/-40$, while the gpt-oss-20b pool gives $+8/-4$ at plurality ≥ 0.75 and $+15/-5$ at ≥ 0.5, corresponding to $+1.3$ points and $p = 0.041$.

Table 15: Risk-controlled threshold selection as in Table 14, with distinct questions instead of (model, question) units as the unit of risk (question-level analysis; supplementary file `ltt_select_q.md`).

answer space	α	selection (development contests)	selected	test questions +/−	test Δ
open	1.0%	all models	vote	— (keep the vote)	+0.00
open	1.0%	held out gpt-oss-20b	vote	— (keep the vote)	+0.00
open	1.0%	held out Qwen3-30B	vote	— (keep the vote)	+0.00
open	1.0%	held out R1-8B	vote	— (keep the vote)	+0.00
open	2.0%	all models	vote	— (keep the vote)	+0.00
open	2.0%	held out gpt-oss-20b	programs, $\theta = 1$, open	2 / 0 (of 123)	+1.63
open	2.0%	held out Qwen3-30B	vote	— (keep the vote)	+0.00
open	2.0%	held out R1-8B	vote	— (keep the vote)	+0.00
open	5.0%	all models	programs, $\theta = 0.5$, open	7 / 3 (of 123)	+1.72
open	5.0%	held out gpt-oss-20b	programs, $\theta = 0.5$, open	4 / 0 (of 123)	+3.25
open	5.0%	held out Qwen3-30B	programs, $\theta = 0.5$, open	2 / 2 (of 123)	−0.27
open	5.0%	held out R1-8B	programs, $\theta = 0.5$, open	4 / 1 (of 120)	+2.22
candidates	1.0%	all models	vote	— (keep the vote)	+0.00
candidates	1.0%	held out gpt-oss-20b	vote	— (keep the vote)	+0.00
candidates	1.0%	held out Qwen3-30B	vote	— (keep the vote)	+0.00
candidates	1.0%	held out R1-8B	vote	— (keep the vote)	+0.00
candidates	2.0%	all models	$\theta = 1$, candidates	2 / 0 (of 123)	+0.38
candidates	2.0%	held out gpt-oss-20b	$\theta = 1$, candidates	1 / 0 (of 123)	+0.81
candidates	2.0%	held out Qwen3-30B	$\theta = 1$, candidates	0 / 0 (of 123)	+0.00
candidates	2.0%	held out R1-8B	$\theta = 0.5$, candidates	1 / 1 (of 120)	−0.50
candidates	5.0%	all models	$\theta = 0.5$, candidates	2 / 3 (of 123)	−0.34
candidates	5.0%	held out gpt-oss-20b	$\theta = 0.5$, candidates	1 / 0 (of 123)	+0.81
candidates	5.0%	held out Qwen3-30B	$\theta = 0.5$, candidates	0 / 2 (of 123)	−1.36
candidates	5.0%	held out R1-8B	$\theta = 0.5$, candidates	1 / 1 (of 120)	−0.50

Thus the failure of unanimous same-model evidence on open-answer mathematics is not universal. GPQA provides a setting in which a rare unanimous contradiction from the same model is informative.

O A Process Reward Model

To apply the decomposition to a standard trained verifier, we scored R1-8B's saved pools with the process reward model Qwen2.5-Math-PRM-7B. Its 4k-token context cannot hold a thinking trace, so each trace is represented by its post-reasoning solution summary, split at paragraph or list boundaries into at most twelve steps plus a final-answer step. For every contested pool, up to eight supporting traces of each top-5 candidate were scored; a trace's reward is the minimum over its steps of the predicted probability that the step is correct, and a candidate's reward is the mean over its scored traces. Scores exist for the mathematics contests with saved R1-8B pools: AIME 2025, HMMT Feb 2026 and AIME 2026 on test (90 units, 129 contested pools with scores) and AIME 2024, HMMT Feb 2025 and BRUMO 2025 on development (90 units, 22 contested pools); 9 and 5 contested pools lack scores and keep their vote.

Two rules use the rewards, both restricted to the candidates, with a tie keeping the vote. *Best-of-N* picks the candidate with the highest reward, and its gate requires that reward to be at least a share θ of the candidates' total reward. The *reward-weighted vote* picks the candidate with the largest product of cluster size and reward, with the same share gate on these products. The decomposition is computed as in Table 21, with every (model, question) unit weighted equally.

On test, neither rule is selective: every selectivity estimate lies within 0.3 points of zero, and the gains and losses follow the accuracy gap. Best-of-N is much less accurate than the vote, by 6.1 points on test and 12.2 on development, so any gate that lets it act loses, and a stricter gate only loses less. The reward-weighted vote rarely disagrees with the vote (at most four overrides on test) and changes little. On development the gated rules show small positive selectivity at $\theta \geq 0.5$, but with only 22 contested pools and at most two overrides the intervals are wide, and it offsets a negative accuracy gap rather than producing a gain. For comparison, R1-8B's own unanimous programs, over all four of its test contests rather than the three scored here, gain $+3.1$ points with

Table 16: Qwen2.5-Math-PRM-7B on R1-8B's saved pools, decomposed as in Table 21 (points; coverage in %; 95% bootstrap intervals over questions). Overrides: contested pools on which the rule changes correctness, with (model, question) units in parentheses. Best-of-N at $\theta = 0.75$ never fires and is omitted.

rule		c	alone	gated	accuracy gap	selectivity	overrides
test (90 units, 129 contested pools)							
best-of-N	$\theta = 0.3$	69.6	-6.1	-4.5	-4.2 [-7.6, -1.2]	-0.3 [-1.8, $+1.4$]	28 (18)
	$\theta = 0.5$	11.3	-6.1	-0.9	-0.7 [-1.5, -0.1]	-0.2 [-1.0, $+0.5$]	4 (4)
reward-weighted vote	$\theta = 0.3$	98.8	$+0.7$	$+0.7$	$+0.7$ [$+0.0$, $+1.9$]	$+0.0$ [$+0.0$, $+0.0$]	4 (2)
	$\theta = 0.5$	69.3	$+0.7$	$+0.4$	$+0.5$ [$+0.0$, $+1.3$]	-0.1 [-0.7, $+0.3$]	2 (1)
	$\theta = 0.75$	29.8	$+0.7$	$+0.0$	$+0.2$ [$+0.0$, $+0.6$]	-0.2 [-0.6, $+0.0$]	0 (0)
development (90 units, 22 contested pools)							
best-of-N	$\theta = 0.3$	90.9	-12.2	-11.1	-11.1 [-17.8, -5.2]	$+0.0$ [-1.4, $+1.4$]	10 (10)
	$\theta = 0.5$	22.7	-12.2	-2.2	-2.8 [-5.7, -0.6]	$+0.6$ [-1.5, $+2.6$]	2 (2)
reward-weighted vote	$\theta = 0.3$	90.9	-4.4	-4.4	-4.0 [-8.5, -0.9]	-0.4 [-1.2, $+0.0$]	4 (4)
	$\theta = 0.5$	54.5	-4.4	-1.1	-2.4 [-4.8, -0.5]	$+1.3$ [-0.7, $+3.4$]	1 (1)
	$\theta = 0.75$	18.2	-4.4	$+0.0$	-0.8 [-1.9, -0.0]	$+0.8$ [$+0.0$, $+1.9$]	0 (0)

selectivity $+3.4$ [$+0.9$, $+6.2$] and an accuracy gap of -0.3. Like any candidate-restricted selector, the PRM is also bounded by the headroom of Table 2.

P Unit-Weighted Precision Under Disagreement and Further Checks

Tables 17, 7 and 8 count overrides per pool, so a question with several seeds counts more than once; Table 17 is the pool-level version of Table 3.

Table 17: Pool-level precision under disagreement (all overrides counted, so questions with several seeds count more than once): overall precision of the confident answers, π_{dis}, ρ_{dis} and $A_{\text{dis}} = \pi_{\text{dis}} - \rho_{\text{dis}}$ with a paired 95% bootstrap interval over distinct questions, contested pools of both splits and all three models. Disagreements: overrides, with (model, question) pairs in parentheses.

source	θ	overall prec.	disagreements	π_{dis}	ρ_{dis}	A_{dis} [95% CI]	rescues / regr.	net
Same-model evidence								
Candidate-conditioned re-solve	0.5	61.8%	142 (64)	27%	25%	$+2$ [-18, $+21$]	$+39$ / -36	$+3$
	0.75	75.0%	27 (13)	48%	22%	$+26$ [-35, $+70$]	$+13$ / -6	$+7$
	1	94.1%	1 (1)	0%	0%	$+0$ [$+0$, $+0$]	$+0$ / -0	$+0$
Candidate verification	0.5	71.1%	87 (47)	32%	33%	-1 [-32, $+26$]	$+28$ / -29	-1
	0.75	77.0%	32 (17)	31%	31%	$+0$ [-46, $+43$]	$+10$ / -10	$+0$
	1	79.5%	12 (9)	17%	50%	-33 [-77, $+20$]	$+2$ / -6	-4
Executed computation								
Own executed programs	0.5	85.1%	106 (44)	66%	13%	$+53$ [$+26$, $+76$]	$+70$ / -14	$+56$
	0.75	90.0%	69 (28)	78%	3%	$+75$ [$+51$, $+97$]	$+54$ / -2	$+52$
	1	95.5%	46 (18)	91%	2%	$+89$ [$+67$, $+100$]	$+42$ / -1	$+41$
Candidate-conditioned programs	0.5	79.4%	140 (58)	59%	10%	$+49$ [$+30$, $+67$]	$+83$ / -14	$+69$
	1	93.6%	39 (15)	79%	5%	$+74$ [$+42$, $+96$]	$+31$ / -2	$+29$
Tool-integrated traces (gpt-oss-20b only)	0.5	95.6%	35 (13)	91%	0%	$+91$ [$+73$, $+100$]	$+32$ / -0	$+32$
	1	100.0%	9 (3)	100%	0%	$+100$ [$+100$, $+100$]	$+9$ / -0	$+9$
Cross-model evidence								
Other-model executed programs	0.5	83.2%	256 (79)	65%	16%	$+49$ [$+25$, $+70$]	$+166$ / -40	$+126$
	1	97.4%	83 (30)	94%	0%	$+94$ [$+81$, $+100$]	$+78$ / -0	$+78$
Other-model reasoning pool	0.5	81.1%	191 (93)	62%	23%	$+39$ [$+24$, $+54$]	$+118$ / -44	$+74$
	1	92.1%	12 (6)	75%	25%	$+50$ [-38, $+100$]	$+9$ / -3	$+6$
Other-model reasoning (8 traces)	0.5	77.4%	87 (69)	55%	22%	$+33$ [$+15$, $+51$]	$+48$ / -19	$+29$
	1	91.2%	6 (6)	83%	0%	$+83$ [$+50$, $+100$]	$+5$ / -0	$+5$

Table 18 recomputes π_{dis}, ρ_{dis} and A_{dis} with every (model, question) unit weighted equally, as in Table 21, with exact Clopper–Pearson intervals on units, which replace percentile intervals that collapse to a point on small cells. It also gives the two covariances whose difference is the selectivity term, $\text{Cov}(g, S) - \text{Cov}(g, V)$ (Section 2.4), and the range of A_{dis} when each contest is left out in turn.

Table 18: Unit-weighted precision under disagreement on the contested pools of the three models. Overrides: pools on which the gate fires and the output differs from the vote, with (model, question) units in parentheses; * fewer than 10 units. π_{dis}, ρ_{dis} in % with exact 95% intervals on units; A_{dis} in points; covariances in points of accuracy, with $\text{Cov}(g, S) - \text{Cov}(g, V)$ the selectivity under the one-sample baseline (Appendix P). Leave one contest out: range of A_{dis}, with the number of contests whose removal changes its sign.

source	rule	overrides	π_{dis}	ρ_{dis}	A_{dis}	$\text{Cov}(g, S)$	$\text{Cov}(g, V)$	leave one out
test								
own programs	$\theta = 0.5$	54 (17)	60 [33, 82]	18 [4, 43]	+42	+3.6	+1.5	[+32, +64] (0)
	$\theta = 0.75$	37 (11)	86 [48, 98]	0 [0, 28]	+86	+3.6	+1.1	[+81, +100] (0)
	$\theta = 1$	25 (8)*	95 [47, 100]	0 [0, 37]	+95	+2.4	+0.6	[+92, +100] (0)
	$\theta = 1$, candidates only	6 (3)*	81 [9, 99]	0 [0, 71]	+81	+1.8	+1.2	[+55, +100] (0)
other models' programs	$\theta = 0.75$	45 (15)	97 [68, 100]	0 [0, 22]	+97	+2.7	+1.2	[+95, +100] (0)
other models' pools	$\theta = 0.75$	17 (10)	65 [35, 93]	0 [0, 31]	+65	+2.2	+1.9	[+52, +89] (0)
8 traces, two other models	$\theta = 0.75$	19 (13)	69 [39, 91]	0 [0, 25]	+69	+2.3	+1.5	[+62, +81] (0)
tool-integrated traces	$\theta = 0.75$	12 (4)*	100 [40, 100]	0 [0, 60]	+100	+3.7	+3.0	[+100, +100] (0)
candidate verification	$\theta = 0.5$	48 (23)	38 [20, 61]	18 [5, 39]	+20	+3.1	+2.4	[−9, +33] (1)
	$\theta = 1$	7 (5)*	18 [1, 72]	55 [5, 85]	−36	+1.4	+1.6	[−57, +0] (0)
	strict	7 (4)*	29 [1, 81]	0 [0, 60]	+29	−0.1	−0.2	[+29, +29] (0)
re-solving	$\theta = 0.5$	68 (24)	21 [10, 47]	12 [3, 32]	+9	+1.5	+1.4	[+3, +15] (0)
development								
own programs	$\theta = 0.5$	52 (27)	66 [46, 83]	14 [2, 29]	+53	+3.7	+1.8	[+48, +56] (0)
	$\theta = 0.75$	32 (17)	74 [50, 93]	9 [0, 29]	+65	+3.1	+1.6	[+60, +100] (0)
	$\theta = 1$	21 (10)	88 [55, 100]	4 [0, 31]	+84	+2.1	+1.0	[+82, +100] (0)
	$\theta = 1$, candidates only	14 (7)*	100 [59, 100]	0 [0, 41]	+100	+2.1	+1.3	[+100, +100] (0)
other models' programs	$\theta = 0.75$	32 (20)	80 [56, 94]	9 [1, 32]	+70	+3.1	+1.6	[+67, +76] (0)
other models' pools	$\theta = 0.75$	26 (16)	79 [54, 96]	18 [4, 46]	+62	+2.0	+1.9	[+58, +71] (0)
8 traces, two other models	$\theta = 0.75$	16 (15)	93 [60, 98]	7 [2, 40]	+86	+2.0	+0.9	[+81, +89] (0)
tool-integrated traces	$\theta = 0.75$	14 (5)*	100 [48, 100]	0 [0, 52]	+100	+2.6	+2.6	[+100, +100] (0)
candidate verification	$\theta = 0.5$	39 (24)	16 [3, 32]	63 [41, 81]	−47	+2.6	+3.0	[−66, −33] (0)
	$\theta = 1$	5 (4)*	43 [1, 81]	43 [7, 93]	+0	+1.6	+1.2	[−100, +40] (2)
	strict	4 (4)*	50 [1, 81]	17 [1, 81]	+33	−0.0	−0.2	[−50, +75] (1)
re-solving	$\theta = 0.5$	74 (40)	29 [15, 44]	31 [21, 52]	−2	+1.5	+0.7	[−12, +3] (2)

Table 19: The eight rules evaluated on the confirmatory contests: development statistics and outcome there (Section 4.3). Development: unit-weighted ρ_{dis} (%; ungated rules not computed), units improved / worsened, and seed-averaged gain. Confirmatory: units improved / worsened and gain. Spearman correlations over the eight rules: development gain with confirmatory gain 0.87 and with confirmatory worsened units 0.15; development worsened units with confirmatory worsened units 0.92 and with confirmatory gain 0.06 (on the six rules with a computed ρ_{dis}, development ρ_{dis} with confirmatory worsened units 0.92).

	development			confirmatory contests	
rule	ρ_{dis}	units +/−	gain	units +/−	gain
programs, candidates only, $\theta = 1$ (pre-registered)	0	7 / 0	+0.9	4 / 0	+0.9
programs, open, $\theta = 1$	4	9 / 1	+1.3	6 / 0	+1.4
programs, open, $\theta = 0.75$	9	13 / 2	+1.8	11 / 0	+2.5
programs, candidates only, $\theta = 0.5$	14	13 / 2	+1.6	7 / 2	+1.1
programs, open, $\theta = 0.5$	14	18 / 4	+2.4	12 / 5	+1.6
CoT–PoT voting, shares	18	24 / 13	+1.6	18 / 10	+1.8
programs alone	–	24 / 16	+1.1	16 / 12	+0.9
8 more traces (re-vote)	–	4 / 3	+0.1	2 / 1	+0.2

Table 20: Two further checks, unit-weighted and with the canonical baseline. Top: eight fresh chain-of-thought traces of the base model (the next sample block, the samples of the "8 more traces" baseline) used as an override source at the programs' gates ($m = 6$); mostly gpt-oss-20b and Qwen3-30B, whose pools have a further sample block. Middle: gpt-oss-20b's eight programs per contested question, executed or with their printed output predicted by the same model without running them (only pools with predictions). Bottom: the $\theta = 0.75$ program override with $n = 16$, 32 and 64 programs per contested question (gpt-oss-20b and Qwen3-30B). π_{dis}, ρ_{dis} in %; gains in points with paired 95% intervals over questions.

source	split	overrides	π_{dis}	ρ_{dis}	alone	gated	accuracy gap	selectivity
8 fresh traces, $\theta = 0.5$	test	11 (11)	18	27	-0.8	-0.1 [-0.5, $+0.3$]	-0.5 [-0.9, 0.0]	$+0.4$ [0.0, $+0.7$]
	dev	24 (21)	7	75	-1.0	-1.2 [-1.8, -0.5]	-0.6 [-1.1, 0.0]	-0.6 [-1.1, -0.2]
8 fresh traces, $\theta = 0.75$	test / dev	0 / 2 (2)	– / 0	– / 100		0.0 / -0.1		
gpt-oss-20b programs, executed, $\theta = 0.5$	test	15 (5)	80	0	$+2.7$	$+3.3$ [$+0.8$, $+6.5$]	$+1.4$	$+1.8$ [$+0.3$, $+3.6$]
	dev	27 (11)	74	15	$+2.6$	$+2.9$ [$+0.2$, $+6.0$]	$+1.5$	$+1.4$ [$+0.1$, $+2.8$]
same programs, output predicted, $\theta = 0.5$	test	0 (0)	–	–	-0.8	0.0	-0.2	$+0.2$
	dev	9 (5)	89	0	$+0.7$	$+1.5$ [$+0.2$, $+3.3$]	$+0.2$	$+1.2$ [$+0.2$, $+2.6$]
same programs, output predicted, $\theta \geq 0.75$	test / dev	0 / 2 (2)	– / 100	– / 0		0.0 / $+0.4$		
programs $n = 16$, $\theta = 0.75$	test	12 (6)	92	0	$+1.9$	$+1.5$ [$+0.3$, $+3.1$]	$+0.6$ [-0.4, $+1.9$]	$+0.9$ [-0.2, $+2.0$]
	dev	25 (10)	88	4	$+2.6$	$+1.9$ [$+0.7$, $+3.4$]	$+1.0$ [$+0.1$, $+2.1$]	$+0.9$ [0.0, $+1.9$]
programs $n = 32$, $\theta = 0.75$	test	15 (7)	73	20	$+2.7$	$+1.1$ [-0.5, $+2.8$]	$+0.9$ [-0.3, $+2.3$]	$+0.2$ [-1.3, $+1.5$]
	dev	29 (12)	76	7	$+1.9$	$+1.8$ [$+0.5$, $+3.4$]	$+0.9$ [-0.1, $+2.1$]	$+1.0$ [0.0, $+2.0$]
programs $n = 64$, $\theta = 0.75$	test	18 (8)	78	0	$+2.6$	$+1.9$ [$+0.3$, $+4.2$]	$+1.0$ [-0.2, $+2.5$]	$+0.9$ [-0.5, $+2.5$]
	dev	29 (12)	76	7	$+2.3$	$+1.8$ [$+0.5$, $+3.4$]	$+0.9$ [0.0, $+2.1$]	$+0.9$ [0.0, $+1.9$]

A matched no-execution control and larger program budgets. Table 20 gives the two checks cited in Section 4.4. At the same budget and gate, the model's own fresh reasoning traces are imprecise when they disagree and lose on development, while its executed programs are precise; the traces almost never agree against the vote at $\theta \geq 0.75$. With more programs, the programs alone become more accurate than the vote, so the decomposition moves part of the gain from selectivity to the accuracy gap, while the programs' precision under disagreement stays high.

Table 21: Decomposition of the test gains into the accuracy-gap and selectivity terms of Eq. (3) (contested pools of all three models; tool-integrated traces: gpt-oss-20b's units only, so their values are not the three-model averages of Table 11). Every (model, question) unit is weighted equally, with its seeds averaged, so for the other sources alone and gated equal the seed-averaged gains of Table 11 (points). Coverage c: weighted share of contested pools on which the gate fires, whether or not the source agrees with the vote (%). Alone: gain of the source replacing the vote whenever at least m samples answer and the top answer is not tied (the gate's own minimum and tie rule), $q - p$. Accuracy gap $= c\,(q - p)$; selectivity = gated − accuracy gap. Paired 95% bootstrap intervals over questions; dev. sel.: selectivity on development. Other models' programs: the 8+8 programs of the two other models; their pools: 64+64 traces; override rules need ≥ 6 answering samples (≥ 32 for pools).

source	rule	c	alone	gated	accuracy gap	selectivity	dev. sel.
own programs	$\theta = 0.5$	39.8	$+0.1$	$+1.7$	$+0.0$ [-0.9, $+1.1$]	$+1.7$ [$+0.4$, $+3.1$]	$+1.7$
	$\theta = 0.75$	28.6	$+0.1$	$+2.2$	$+0.0$ [-0.6, $+0.9$]	$+2.2$ [$+0.9$, $+3.6$]	$+1.3$
	$\theta = 1$	17.2	$+0.1$	$+1.6$	$+0.0$ [-0.4, $+0.5$]	$+1.6$ [$+0.4$, $+2.9$]	$+1.0$
tool-integrated traces	$\theta = 0.75$	46.5	$+4.1$	$+3.3$	$+1.9$ [$+0.4$, $+4.1$]	$+1.4$ [-0.3, $+3.1$]	$+1.2$
other models' programs	$\theta = 0.75$	29.1	$+3.4$	$+2.7$	$+1.0$ [$+0.0$, $+2.4$]	$+1.7$ [$+0.4$, $+3.2$]	$+1.4$
other models' pools	$\theta = 0.75$	24.9	$+2.1$	$+0.8$	$+0.5$ [$+0.1$, $+1.0$]	$+0.3$ [-0.2, $+0.9$]	$+0.1$
8 traces, two other models	$\theta = 0.75$	23.8	$+1.1$	$+1.0$	$+0.3$ [-0.1, $+0.6$]	$+0.8$ [$+0.1$, $+1.5$]	$+1.1$
candidate verification	$\theta = 1$	19.0	$+0.1$	-0.2	$+0.0$ [-0.3, $+0.3$]	-0.2 [-0.7, $+0.3$]	$+0.4$
candidate-conditioned re-solve	$\theta = 0.5$	67.6	$+0.5$	$+0.5$	$+0.3$ [-0.5, $+1.2$]	$+0.2$ [-0.2, $+0.6$]	$+0.7$

Is the gate a problem-type router? A keyword classifier ("how many", "number of", "count", ...) marks 124 of 366 test and 153 of 549 development units as counting questions. Overriding with the programs' plurality on every marked question (≥ 6 answering programs, no agreement threshold) overrides 53 (18 units) and 29 (18) pools, with π_{dis} 42% and 55% against ρ_{dis} 30% and 18%, and gains $+0.5$ [-1.3, $+2.5$] and $+1.1$ [-0.4, $+2.9$] points. The unanimous rule gains $+0.9$ on marked and $+0.7$ on unmarked test questions ($+0.6$ and $+0.7$ on development), with π_{dis} of

100% and 89% (100% and 81%). Its precision therefore comes from the programs' agreement and holds whether or not the question is a count.

Table 22: Decomposition on the five confirmatory contests (441 units, one run per model), as in Table 21: coverage c (%), gains in points with paired 95% intervals over questions. Per model (one-sample baseline), the unanimous rule's selectivity on test is $+3.4$ (R1-8B), $+1.1$ (gpt-oss-20b) and $+0.6$ (Qwen3-30B), and on development $+1.1$, $+0.5$ and $+1.3$.

rule	c	alone	gated	accuracy gap	selectivity
programs, open, $\theta = 1$	9.9	$+1.4$	$+1.4$ [$+0.2$, $+2.7$]	$+0.1$ [-0.1, $+0.5$]	$+1.2$ [$+0.2$, $+2.4$]
programs, open, $\theta = 0.75$	18.5	$+1.4$	$+2.5$ [$+0.7$, $+5.0$]	$+0.3$ [-0.2, $+0.9$]	$+2.2$ [$+0.8$, $+4.1$]

Table 23: The 32 rules of the regression-transfer analysis (Section 4.3; Table 11 shows a subset), grouped into 13 source families: seed-averaged gain (points) and (model, question) units improved / worsened, on development and test. Spearman correlations over the 32 rules, with 95% intervals from a bootstrap that resamples the 13 families: development gain with test gain 0.71 [0.32, 0.87]; development worsened units with test worsened units 0.88 [0.79, 0.94], and 0.54 [0.27, 0.77] controlling for development coverage (units changed); the worsened share of changed units, development with test, 0.67 [0.40, 0.87]; development gain with the test worsened share 0.09 [-0.32, $+0.57$] (a rule that changes nothing counts as share 0; excluding the two such rules, re-solving and cross traces at $\theta = 1$, gives 0.64 and 0.04). With one rule per family (13 rules), the worsened share carries over at 0.74 [0.30, 0.94] (loosest rule of each family) and 0.63 [0.14, 0.93] (strictest), and development gain predicts it at -0.03 and $+0.29$ (intervals covering zero).

family	rule	dev. gain	dev. +/−	test gain	test +/−
exec open	programs alone	$+1.1$	24/16	-0.9	13/17
exec open	programs, $\theta = 0.5$	$+2.4$	18/4	$+1.7$	10/3
exec open	programs, $\theta = 0.75$	$+1.8$	13/2	$+2.2$	9/0
exec open	programs, $\theta = 1$	$+1.3$	9/1	$+1.6$	7/0
exec candidates	programs, cand., $\theta = 1$ (pre-reg.)	$+0.9$	7/0	$+0.4$	2/0
exec candidates	programs, cand., $\theta = 0.5$	$+1.6$	13/2	-0.3	2/3
exec candidates	programs, cand., $\theta = 0.75$	$+1.2$	10/1	$+0.4$	2/0
hinted programs	hinted programs, $\theta = 1$	$+0.6$	5/2	$+1.5$	6/0
re-solving	re-solving alone	-1.2	11/22	$+0.6$	7/4
re-solving	re-solving, $\theta = 0.5$	-0.1	11/15	$+0.5$	6/3
re-solving	re-solving, $\theta = 0.75$	-0.2	1/3	$+0.5$	3/0
re-solving	re-solving, $\theta = 1$	$+0.0$	0/0	$+0.0$	0/0
verification	verification alone	-2.3	6/25	$+0.0$	9/9
verification	verification, $\theta = 1$	$+0.0$	1/2	-0.2	1/2
8 cross traces	8 cross traces alone	$+1.8$	29/26	$+1.2$	19/17
8 cross traces	8 cross traces, $\theta = 0.5$	$+2.0$	21/9	$+1.0$	14/8
8 cross traces	8 cross traces, $\theta = 0.75$	$+1.5$	12/1	$+1.0$	9/0
8 cross traces	8 cross traces, $\theta = 1$	$+0.4$	5/0	$+0.0$	0/0
other programs	other models' programs alone	$+2.5$	34/21	$+4.3$	30/13
other pools	other models' pools alone	$+3.8$	41/19	$+2.1$	23/11
tool-integrated	TIR alone (gpt-oss)	$+2.0$	15/1	$+1.9$	9/0
tool-integrated	TIR, $\theta = 0.5$	$+1.0$	7/0	$+1.4$	5/0
tool-integrated	TIR, $\theta = 0.75$	$+0.9$	5/0	$+1.1$	4/0
tool-integrated	TIR, $\theta = 1$	$+0.2$	1/0	$+0.5$	2/0
CoT-PoT	CoT–PoT, counts	$+0.5$	8/2	$+0.9$	7/0
CoT-PoT	CoT–PoT, shares	$+1.6$	24/13	$+2.6$	19/6
CoT-PoT	CoT–PoT, max	$+1.6$	21/13	$+1.6$	16/8
CoT-PoT	CoT–PoT, agreement	$+1.4$	25/15	$+1.7$	18/9
jury	jury, shares	$+3.8$	34/13	$+3.0$	22/5
jury	jury, votes	$+3.0$	22/2	$+1.9$	14/2
share-sum	share-sum, programs	$+0.9$	20/10	$+0.2$	8/4
8 more traces	8 more traces (re-vote)	$+0.1$	4/3	$+0.3$	3/0

Where the gate fires. For own programs, other models' programs and the eight cross-model traces, the gate covaries more with the source being right than with the vote being right, so their selectivity is positive (Table 21). For verification and re-solving the two covariances are nearly equal: the gate fires about as often where the vote is right as where the source is, which is what a confidence that tracks easy questions predicts. Other models' pools and tool-integrated traces also have nearly equal covariances; their gains come from the accuracy gap.

Table 24: The lens on standard and stronger correction mechanisms, decomposed as in Table 21 (test points, unit-weighted; paired 95% intervals over questions). Top: contested pools of all three models; the jury and CoT–PoT voting have no gate. Bottom: R1-8B's saved pools of AIME 2025, HMMT Feb 2026 and AIME 2026 (90 units; development: AIME 2024, HMMT Feb 2025, BRUMO 2025), where stronger judges and a process reward model chose among the top-5 candidates with 8 samples (PRM: reward share); R1-8B's own programs on the same pools. Qwen3-30B-A3B verification covers 75 of the 138 test and 14 of the 27 development contested pools; missing pools keep the vote. Dev. sel.: selectivity on development (gated gains there: jury $+3.8$, CoT–PoT $+1.6$, own programs $+1.3$; on R1-8B's pools -1.1 for both selection judges, 0.0 for Qwen3 verification and own programs).

mechanism	rule	alone	gated	accuracy gap	selectivity	dev. sel.
all three models, contested pools						
3-model jury, summed shares	no gate	$+3.0$	$+3.0$	$+3.0$	0	0
CoT–PoT voting, shares	no gate	$+2.6$	$+2.6$	$+2.6$	0	0
verification (base model)	margin	$+0.0$	$+0.3$	$+0.0$	$+0.3$ $[-0.6, +1.4]$	-1.9
own programs	$\theta = 1$	$+0.1$	$+1.6$	$+0.0$	$+1.6$ $[+0.4, +2.9]$	$+1.0$
R1-8B's saved pools (90 test and 90 development units)						
GenSelect-style, gpt-oss-20b	$\theta = 0.75$	$+2.6$	$+2.3$	$+0.6$	$+1.6$ $[+0.1, +3.6]$	-0.3
GenSelect-style, gpt-oss-120b	$\theta = 0.75$	$+5.7$	$+4.9$	$+2.4$	$+2.4$ $[+0.7, +4.4]$	-1.1
GenSelect-style, Qwen3-30B-A3B	$\theta = 0.75$	$+0.6$	$+1.1$	$+0.3$	$+0.8$ $[-0.7, +2.5]$	$+0.2$
verification, Qwen3-30B-A3B	margin ≥ 0.5	$+0.1$	$+0.2$	$+0.0$	$+0.2$ $[-0.2, +0.6]$	$+0.2$
PRM best-of-N	share ≥ 0.5	-6.1	-0.9	-0.7	-0.2 $[-1.0, +0.5]$	$+0.6$
R1-8B own programs (same pools)	$\theta = 1$	$+0.4$	$+4.1$	$+0.1$	$+4.0$ $[+0.9, +7.1]$	$+0.1$

Table 25: Precision under disagreement by the vote's share of its 64-trace pool, and vote-margin gates for rules without a source-agreement gate (unit-weighted; A_{dis} in points, units improved / worsened).

rule	vote share	test A_{dis}	test +/−	dev. A_{dis}	dev. +/−
programs, $\theta = 0.75$	< 0.5	$+81$	6/0	$+54$	9/2
	0.5–0.65	$+100$	3/0	$+100$	5/0
	0.65–0.8	$+100$	2/0	–	0/0
verification, $\theta = 0.5$	< 0.5	$+39$	8/2	-62	1/10
	0.5–0.65	-25	2/2	-6	3/6
re-solving, $\theta = 0.5$	< 0.5	$+13$	4/2	-5	8/11
	0.5–0.65	$+3$	2/2	$+4$	3/4
CoT–PoT voting, all contested	–	$+26$	19/6	$+15$	24/13
only where vote share < 0.4		$+26$	12/3	$+16$	15/7
programs alone, all contested	–	-10	13/17	$+10$	24/16
only where vote share < 0.4		$+30$	9/3	$+19$	13/6

Contest dependence. Leaving out any one contest never changes the sign of A_{dis} for own programs, other models' programs, the eight cross-model traces or other models' pools at $\theta = 0.75$, on either split. The sign changes occur for verification (at $\theta = 0.5$ on test, at $\theta = 0.75$ on both splits and at $\theta = 1$ on development) and for re-solving at $\theta = 0.5$ on development, apart from two cells with three or four units (other models' pools at $\theta = 1$ and strict verification on development).

Verification with a margin gate. The margin rule of Table 3 outputs the candidate with the most "correct" verdicts when the vote's candidate has at least k "incorrect" verdicts and that candidate at least k "correct" ones. With $k = 2$ it overrides 71 pools (25 units) on test and 44 (30) on development; its gated gain is $+0.3$ [-0.7, $+1.6$] and -2.2 [-3.6, -0.9] points, with selectivity $+0.3$ [-0.6, $+1.4$] and -1.9 [-3.1, -0.8]. With $k = 3$ it gains $+0.7$ [0.0, $+1.7$] and -0.6 [-1.3, $+0.1$], and with $k = 4$ it fires on under 1% of contested pools. No margin makes verification selective on both splits.

Published combination rules. Share-normalized CoT–PoT voting over the pool and the eight programs changes 116 pools (45 units) on test and 123 (67) on development, with π_{dis} of 40% and 33% against ρ_{dis} of 13% and 18%; it gains $+2.6$ [-0.1, $+5.6$] and $+1.6$ [-0.5, $+3.8$] points, all of it accuracy gap, since it has no gate. The three-model jury with summed shares gains $+3.0$ [$+1.1$, $+5.2$] and $+3.8$ [$+2.1$, $+5.8$]; for R1-8B it gains $+7.0$ and $+8.2$, and for gpt-oss-20b and Qwen3-30B between $+0.5$ and $+1.8$. Both are more accurate predictors than the vote, which is a different claim from being selective.

Stronger judges on R1-8B's saved pools. The judges of Table 24 were run on the contested pools of R1-8B's saved runs of three test contests (AIME 2025, HMMT Feb 2026 and AIME 2026: 138 contested pools of 90 questions) and three development contests (AIME 2024, HMMT Feb 2025 and BRUMO 2025: 27 contested pools, one run each), with the pool's top-5 candidates in random order and 8 samples per question (verification: 4 verdicts per candidate). A pool without a judge's record keeps the vote for that judge (8–9 of the 138 test and 2–5 of the 27 development pools for the selection, re-solving and reward-model judges; 63 and 13 for Qwen3-30B-A3B verification, which was not completed); R1-8B's own programs are also reported on exactly the pools covered by gpt-oss-20b's selection, with the same result. With the canonical baseline (at least six answering samples, as for the gates), GenSelect-style selection by gpt-oss-20b gains $+2.6$ [$+0.4$, $+5.8$] points used alone on test; with any number of answering samples, $+6.0$ [$+2.3$, $+10.6$], and -3.3 [-7.8, 0.0] on development. Selection by Qwen3-30B-A3B gains $+0.6$ alone on test and -2.2 on development. Re-solving by the stronger models, with answers mapped to the candidates, behaves in the same way. The development sets are small (27 contested pools), so these development contrasts are descriptive.

Verification with a stricter rule. Requiring the vote to be rejected by all of its verdicts and one other candidate to be accepted by all of its (Appendix B) leaves the conclusion unchanged: the rule fires on about 1% of contested pools, overrides 7 and 4 pools on test and development, and gains $+0.1$ points on each split.

Candidate-restricted programs. Restricted to the pool's top-5 candidates, program overrides remain precise, with π_{dis} of 81% on test and 100% on development under unanimity, and keep positive selectivity: $+0.6$ [$+0.1$, $+1.2$] and $+0.8$ [$+0.3$, $+1.5$] points. They gain less than the open rule ($+0.4$ and $+0.9$) because they override far less often, not because they are less precise; candidate verification, also restricted to the candidates, has π_{dis} of 18% and 43% under unanimity.

Sensitivity of the split to its conventions. Table 21 uses the canonical baseline: the source alone applies the gate's own minimum m and tie rule. Under it the own programs' selectivity is $+1.6$ [$+0.4$, $+2.9$] points on test and $+1.0$ [$+0.4$, $+1.7$] on development under unanimity, and $+2.2$ [$+0.9$, $+3.6$] and $+1.3$ [$+0.4$, $+2.3$] at $\theta = 0.75$. If the source alone needs only one answering sample, the minimum m counts as selectivity and these rise to $+1.8$, $+1.1$, $+2.5$ and $+1.5$. If $\mathcal{D}$ is every pool, settled ones included, programs used alone lose -5.1 points [-9.2, -1.0] on test and -1.5 on development while the unanimous rule gains $+1.9$ and $+1.3$, so the accuracy gap is negative and selectivity exceeds the gain. Under the one-sample convention a rule without a gate can show spurious selectivity when it cannot act on every pool: the three-model jury then shows

$+0.3$ [$+0.1$, $+0.5$], because R1-8B's extra runs have no partner pools, so the rule abstains exactly where the source is not available ($\mathrm{Cov}(g, S - V) = (1 - c)(q - p) > 0$). Restricting $\mathcal{D}$ to the pools where an ungated rule can act removes this term.

Q Qualitative Program Analysis

Rescues. The four R1-8B test problems rescued by the unanimous open execution rule are finite enumerations: ordered triples (a, b, c) of positive integers $\leq 3^6$ satisfying $a^3+b^3+c^3 \equiv 0 \pmod{3^7}$ (AIME 2025, all three seeds; pool 147, programs 735); $\sum_k k\binom{2k}{k}2^k \bmod 2027$ (HMMT Feb 2026, four of six seeds; pool 0, programs 289); sequences of positive integers summing to 12 under a parity-dependent operation; and 7-tuples over $\{1, 2, 3\}$ satisfying two divisibility constraints (AIME 2026, all five seeds; 157 and 393).

The correct value lies outside the pool's top five in 15 of these 17 pools. gpt-oss-20b's test rescues are the same AIME 2025 problem—its contested pool also leads with 147—and an HMMT Nov 2025 count, where the program answer is 81,920 against the vote's 20,480, a runner-up candidate, each in all three seeds. Qwen3-30B's test rescue is the AIME 2026 problem with answer 393 in one seed.

Every override was audited manually against the reference answer (Table 30). Two apparent rescues, votes of 74° and 55° against references 74 and 55 (gpt-oss-20b, HMMT Feb 2026 question 26 and SMT 2025 question 22; the first under $\theta = 0.75$, the second under $\theta = 0.75$ and the unanimous rule), were grading artifacts of degree marks; the grader was corrected before the reported tables were produced, and neither counts in any table.

Silence. On R1-8B's three development contests with saved pools (AIME 2024, HMMT Feb 2025 and BRUMO 2025; one seed), the unanimous program override never fires. Of 27 contested pools, 16 have a correct vote. On the 11 wrong ones, the eight programs are correct zero times on eight pools, once on two, and seven times on one, a grid-path count where one faulty depth-first search breaks unanimity (the plurality rule and both other models' programs rescue it). The other ten questions involve geometry with irrational answers, continuous probability and expectations, a closed-form count, or counting problems that the programs mis-encode. SMT 2025, the fourth R1-8B development contest without a rescue, has one new seed and is not part of this audit.

When programs share the pool's *interpretation* error, as in an expected-value problem with a stopping rule from HMMT Feb 2026, they are unanimous on the same wrong answer. The rule therefore remains silent rather than introducing a regression. Execution can separate some computational errors from ordinary reasoning, but it does not provide independent evidence about whether the model interpreted the problem correctly.

Program failure modes. gpt-oss-20b occasionally emits a tool call instead of a program (0.2% after an explicit no-tools instruction, executed as the intended program). Under the initial lenient prompt, Qwen3-30B printed a literal without computation in 29% of samples. Such programs are discarded, and the strict prompt in Appendix V is used for all reported experiments. Approximately 1–1.5% of programs time out.

R Exploratory versus confirmatory evidence, regression bounds and claim map

Evidence by status. Table 26 lists the program-override results by what had been seen when the rule was fixed (Appendix C). The rule was identified on the R1-8B pools; gpt-oss-20b and Qwen3-30B were scored afterwards with every rule and threshold fixed; ArXivMath 2026 was run after all analyses. After all analyses above were complete we ran the frozen rules once more on five contests no analysis had used (HMMT February 2023 and 2024, AIME 2023 I and II, and the August 2026 release of ArXivMath, which is disjoint from the January–June 2026 releases used in Appendix T; 147 questions, all three models, one seed of the same adaptive protocol and eight programs per contested question). All three models completed the run. The rules evaluated on it were fixed before it was sampled (unanimity, $\theta \in \{0.5, 0.75\}$, the candidate-restricted primary at $\theta = 1$ and

Table 26: Program-override results by evidence status. Units: (model, question) pairs improved / worsened; questions: distinct problems improved / worsened; Δ: seed-averaged gain over the vote with a 95% bootstrap CI over questions.

		programs, $\theta = 1$, open			override $\theta = 0.75$		
scope	status	units +/−	questions +/−	Δ [95% CI]	units +/−	questions +/−	Δ [95% CI]
R1-8B, test contests	rule identified on these pools	4/0	4/0	+3.1 [+0.6, +6.4]	4/0	4/0	+3.1 [+0.6, +6.4]
R1-8B, development contests	rule identified on this model	2/0	2/0	+1.1 [+0.0, +2.7]	5/1	5/1	+2.2 [+0.0, +4.9]
gpt-oss-20b, test contests	model held out when the rule was fixed	2/0	2/0	+1.6 [+0.0, +4.1]	3/0	3/0	+2.4 [+0.0, +5.7]
gpt-oss-20b, development contests	model held out	4/1	4/1	+1.3 [−0.0, +3.1]	5/1	5/1	+1.8 [+0.2, +4.0]
Qwen3-30B, test contests	model held out	1/0	1/0	+0.3 [+0.0, +0.8]	2/0	2/0	+1.1 [+0.0, +3.0]
Qwen3-30B, development contests	model held out	3/0	3/0	+1.5 [+0.0, +3.5]	3/0	3/0	+1.5 [+0.0, +3.5]
held-out models, test contests	pooled	3/0	3/0	+0.9 [+0.0, +2.2]	5/0	4/0	+1.8 [+0.1, +4.1]
held-out models, both splits	pooled	10/1	10/1	+1.2 [+0.4, +2.1]	13/1	12/1	+1.7 [+0.7, +2.8]
gpt-oss-20b, ArXivMath 2026	contest held out (question-disjoint)	1/0	1/0	+0.5 [+0.0, +1.4]	2/0	2/0	+0.9 [+0.0, +2.3]
R1-8B, confirmatory contests	run after all rules were fixed	1/0	1/0	+0.7 [+0.0, +2.0]	3/0	3/0	+2.0 [+0.0, +4.8]
gpt-oss-20b, confirmatory contests	run after all rules were fixed	4/0	4/0	+2.7 [+0.7, +6.1]	6/0	6/0	+4.1 [+1.4, +7.5]
Qwen3-30B, confirmatory contests	run after all rules were fixed	1/0	1/0	+0.7 [+0.0, +2.0]	2/0	2/0	+1.4 [+0.0, +3.4]
all models, confirmatory contests	pooled	6/0	5/0	+1.4 [+0.2, +2.7]	11/0	7/0	+2.5 [+0.7, +5.0]

$\theta = 0.5$, programs alone, share-normalized CoT–PoT voting and eight more traces: eight rules), with one seed per model, so a (model, question) unit is one pool. Neither strict override worsens any of the 441 units: the unanimous program override improves 6 units on 5 distinct questions ($+1.4$ points [$+0.2$, $+2.7$]; unit-level sign $p = 0.031$, anti-conservative because units share questions; question-level tests are in Section 4.4), and the $\theta = 0.75$ override 11 units on 7 questions ($+2.5$ [$+0.7$, $+5.0$]; $p = 0.001$), with gains on every model (gpt-oss-20b $+4.1$, R1-8B $+2.0$, Qwen3-30B $+1.4$). The looser rules again trade rescues for regressions: $\theta = 0.5$ improves 12 units and worsens 5 ($+1.6$ [-0.7, $+4.3$]), programs alone 16 and 12 ($+0.9$), share-normalized CoT–PoT voting 18 and 10 ($+1.8$ [-1.1, $+5.0$]), and the candidate-restricted primary rule 4 and 0 ($+0.9$); eight more traces gain $+0.2$. The pattern of Table 11 therefore holds on questions no analysis had seen, and the $\theta = 0.75$ override, examined on the test contests only after their results were known, is confirmatory evidence here. Two caveats: every strict-rule rescue is on HMMT February 2023, HMMT February 2024 or AIME 2023 II, which predate the models' releases, and none on AIME 2023 I or ArXivMath August 2026; and six of the seven rescued problems are finite counts (tilings, reachable values, permutation triples, digit permutations, subset families, a residue condition), the seventh a triangle length. Post-release integer-answer contests would be the natural next confirmatory set; the 2026 BRUMO, SMT and CMIMC sets were not publicly available when this run was made.

Held-out models. gpt-oss-20b and Qwen3-30B were sampled after the unanimous open rule was fixed on R1-8B, so their units on every split test a fixed rule. Counting distinct questions (net change over the two models) and excluding every problem the rule rescued for R1-8B (AIME 2025 q14 and AIME 2026 q29), it improves 12 questions (development: BRUMO 2025 q20, CMIMC 2025 q0, q4, q14, q33, HMMT February 2025 q13, SMT 2025 q38; test: HMMT November 2025 q5; confirmatory: HMMT February 2023 q16, HMMT February 2024 q18 and q28, AIME 2023 II q10) and worsens one (CMIMC 2025 q6, the disputed key), two-sided sign test $p = 0.003$ (14 and 1, $p = 0.001$, without the exclusion). The $\theta = 0.75$ rule, adopted later, gives 12 and 1 under the same exclusion.

Regression probability conditional on overriding. For an override rule the deployment risk is ρ_{dis}, the probability that an override worsens the answer. Over both splits the unanimous program override overrides 46 contested pools on 18 (model, question) units and regresses on 1 pool[1] (2.2%; exact 95% Clopper–Pearson interval up to 12% per pool, up to 27% per (model, question) unit); the $\theta = 0.75$ override overrides 69 pools on 28 (model, question) units with 2 regressions (up to 10% per pool, 24% per unit); $\theta = 0.5$ regresses on 14 of 106 (13.2% [21% upper]). On test alone the strict rules have 0 regressions in 25 and 0 in 37 overrides (upper bounds 14% and 9% per pool, 37% and 28% per unit). Per-pool bounds treat overlapping runs of a question as independent and are descriptive; the per-unit bounds, not the observed zeros, are the honest statement of the rule's safety.

Multiplicity. Table 11 compares 20 rules besides the pre-specified one. Holm-adjusted sign tests over (model, question) units on test give $p = 0.074$ for the $\theta = 0.75$ override, $p = 0.030$ for the

[1] This regression is gpt-oss-20b on CMIMC 2025 q6 (problem 7 in the contest's numbering), a sum of 3-adic valuations over $n \leq 810$. Exact enumeration gives 932, which the programs of two model families output; the published key is 930. We keep the published key, so this override counts as a regression in every table.

jury with summed shares and $p = 0.219$ for share-normalized CoT–PoT voting (the supplementary file `qstats_extra.md` lists all).

No-execution control. Appendix S reports the same program samples with the answer the model states in its reasoning before writing the code, so that execution is the only difference.

Claim map. Table 27 maps each claim of the paper to the analysis and the generated file that supports it.

Table 27: Claims, supporting analyses and the result files (supplement) they are generated from.

claim	analysis	result file
most wrong contested votes lack the answer among candidates (§4.1)	Table 2; K varied and per-unit shares in Table 9	`*_s3_contested.md`, `pool_flags.json`
same-model verification / re-solving: no gain (§4.1)	Table 11 rows; π_{dis} in Table 3	`allrules_*.md`, `calibration_*.md`
disagreement advantage separates sources (§4.2)	Table 3, Appendices D and E	`calibration_contested*.md`, `adv_curve.md`
no source is independent of the vote (§4.3)	Table 10 (Appendix G)	`independence_*.md`
the unanimous program override gains on test with no observed regression (§4.4)	Table 11; Table 26; regression bounds above	`allrules_test_contested.md`, `qstats_extra.md`
most rescues are non-candidate answers (§4.4)	candidate-restricted rows of Table 11; Appendices I and Q	`calibration_contested.md`, `changes_*_egar.txt`
execution, not prompting, carries the gain (Appendix S)	Table 29: stated-answer control, other-model traces, tool-integrated traces	`stated_*.md`, `calibration_stated_*.md`
rule selection by development gain does not transfer (Appendix L)	Table 13	`select_lomo.md`
risk-controlled selection is exploratory (Appendix M)	Tables 14 and 15	`ltt_select.md`, `ltt_select_q.md`
results replicate with independent programs per seed (Appendix U)	Table 32, Table 33	`perseed_*.md`, `pot*_*.md`
question-disjoint contest (Appendix T)	Table 31	`heldout*.md`
a process reward model is not selective (§4.3)	Table 16 (Appendix O)	`prm_decomp.md`
unit-weighted π_{dis}, covariances, contest dependence (§4.2)	Table 18 (Appendix P)	`review_extras.md`
cost (§4.4)	Table 34 (Appendix W)	`cost*.md`

Inventory. Table 28 gives the exact numbers of distinct questions, pools and contested pools per model and split. Every question has eight programs; every question contested in some seed has eight hinted re-solves, eight hinted programs and four verification samples per candidate; gpt-oss-20b and Qwen3-30B additionally have 56 more programs per contested question (Appendix U) and gpt-oss-20b eight tool-integrated traces per contested question.

Table 28: Data inventory (contested pools: complete pools whose top share is below 0.8 after 64 traces).

model	split	questions	pools	contested pools
R1-8B	dev	183	183	66
R1-8B	test	120	443	151
gpt-oss-20b	dev	183	549	220
gpt-oss-20b	test	123	369	142
Qwen3-30B	dev	183	549	122
Qwen3-30B	test	123	369	88

S WHAT EXECUTION CONTRIBUTES

Does the unanimous program override gain from execution, or merely from a different prompt? Table 29 holds the samples fixed and varies only what is read from them. *Stated answer*: the same eight program samples, but the evidence is the answer the model states in its reasoning before it writes the code (the last "the answer is x", "ANSWER: x" or boxed value in the text preceding the code block; found in 89% of samples, the rest abstain); no code is run. *Executed answer*: the interpreter's output, as in the paper. The other rows are the matched-budget controls of Table 11: eight more traces of the same model, four traces from each of the two other models, and, for gpt-oss-20b, eight tool-integrated traces. The program prompt alone, read before any code, does not carry the gain; whether a finished program is precise without being run is tested only by the small predicted-output control (Table 20). Read without execution, the same samples are a poor source: their stated answers agree with the executed answers on only 57% of the samples that have both, and where both exist the stated answer is right 47% of the time against 76% for the executed one. As an override source the stated answers have a precision under disagreement of 5% at a simple majority and 4% at $\theta = 0.75$ (executed: 66% and 78%), a negative disagreement advantage at every threshold, and they are unanimous against the vote on only 8 pools; overriding with them loses -2.9 points on test at $\theta = 0.5$ and -1.2 at $\theta = 0.75$, and replacing the vote by their plurality loses -6.5. The stated answer is a mid-reasoning belief rather than a finished chain of thought, so

it is a lower bound on what the program-writing prompt contributes without execution; the eight additional chain-of-thought traces ($+0.3$ on test) are the upper bound, and both are far from the executed programs. Tool-integrated traces, where the model executes as it reasons, are more precise still at a similar token cost (about 100k output tokens per contested question for eight, against 135k for eight programs) but were available for one model. Finally, the audit below suggests a post-hoc filter: discarding a program whose printed answer appears as a literal in its own print statement (8.4% of R1-8B's answering programs, 2.1% of Qwen3-30B's, 0.9% of gpt-oss-20b's) leaves the unanimous program override unchanged (7/0 test and 9/1 development units; the one development regression, CMIMC 2025 q6, comes from faithful programs, see the program audit below), turns the $\theta = 0.75$ override's development record from 13/2 into 13/1 and the $\theta = 0.5$ override's test record from 10/3 into 10/2. We report it as an audit-motivated variant, not as part of the frozen rule.

Table 29: Execution attribution: the same program samples read without and with execution, next to the trace-based controls. Overrides on contested pools of both splits, with (model, question) pairs in parentheses; π_{dis}, ρ_{dis} and A_{dis} as in Table 3; test and development gains as in Table 11.

source	θ	overrides (units)	π_{dis}	ρ_{dis}	A_{dis}	test Δ [95% CI] (units +/−)	dev Δ (units)
8 more traces, same model	—	—	—	—	—	+0.3 [+0.0, +0.6] (+3/−0)	+0.1 (4/3)
8 program samples, stated answer (no execution)	0.75	56 (31)	4%	48%	−44	−1.2 [−2.4, −0.3] (0/6)	−1.8 (2/11)
8 program samples, stated answer (no execution)	1	8 (4)	25%	38%	−13	+0.0 [+0.0, +0.0] (0/0)	−0.1 (2/1)
8 program samples, executed answer	0.75	69 (28)	78%	3%	+75	+2.2 [+0.5, +4.2] (9/0)	+1.8 (13/2)
8 program samples, executed answer	1	46 (18)	91%	2%	+89	+1.6 [+0.3, +3.3] (7/0)	+1.3 (9/1)
8 traces, 2 other models	0.75	35 (28)	80%	6%	+74	+1.0 [+0.3, +1.9] (9/0)	+1.5 (12/1)
8 traces, 2 other models	1	6 (6)	83%	0%	+83	+0.0 [+0.0, +0.0] (0/0)	+0.4 (5/0)
8 tool-integrated traces (gpt-oss-20b only)	0.75	26 (9)	100%	0%	+100	+1.1 [+0.3, +2.2] (4/0)	+0.9 (5/0)
8 tool-integrated traces (gpt-oss-20b only)	1	9 (3)	100%	0%	+100	+0.5 [+0.0, +1.4] (2/0)	+0.2 (1/0)

Program audit. Table 30 classifies, by hand, every program behind the 30 (model, question) units on which the open program override fires under unanimity or at $\theta = 0.75$ (both splits, 240 programs): A, the program genuinely computes the answer from the problem's objects (enumeration, search, simulation, exact arithmetic over the stated structure); B, it evaluates a closed-form expression the model derived in its reasoning; C, it prints a hard-coded value on its answer line, with any computation decorative or dead; D, it runs a computation that misreads the problem. Overall 178 programs are A (74%), 19 B (8%), 39 C (16%) and 4 D (2%); on 24 of the 30 units the override output is produced by a majority of category-A programs, and these include every rescued unit but three (BRUMO 2025 q24 and SMT 2025 q38, where hard-coded roots happen to be right, and the reference-form rescues). The audit locates the failure mode of the looser rule: the regression of the $\theta = 0.75$ override on R1-8B's CMIMC 2025 q19 and four of the five wrong-but-harmless overrides come from category-C programs that print a literal after dead code—a pattern the sandbox's triviality check (a program whose only statement prints a literal) does not catch—or from six programs evaluating the same wrong closed form. The remaining wrong override (CMIMC 2025 q6) is unanimous across seven faithful big-integer implementations for gpt-oss-20b and eight for Qwen3-30B, all of which return 932 against the reference 930; exact enumeration also gives 932, so the published key appears to be wrong. In one gpt-oss-20b seed the vote matched the key, and that pool is the unanimous rule's only regression in every table (we keep the published key). A post-hoc filter that discards a program whose printed answer appears as a literal in its own print statement is evaluated at the end of the text accompanying Table 29 above; it is audit-motivated and not part of the frozen rule. The same audit of the six confirmatory rescues (Appendix R) finds 44 answering programs, none of which prints a hard-coded answer or contains the reference value in its source; all compute the answer, by enumeration, search or exact symbolic algebra. Nor do the models recall these answers: on the six rescued questions, 1, 1, 3, 10, 1 and 12 of the model's own 64 traces reach the reference.

T QUESTION-DISJOINT HELD-OUT CONTEST

After the rules had been fixed, gpt-oss-20b was run on ArXivMath 2026 (MathArena; questions drawn from 2026 arXiv papers, none of which any analysis in this paper had used) with one seed of the same adaptive protocol and eight programs per contested question. Table 31 reports the program rules of Table 11 that need no second model. The contest is far harder than the nine of the main study: the vote is correct on 20.2% of its 213 questions, 179 pools are contested, and of the 152

Table 30: Hand classification of the 240 programs behind the 30 units on which the open program override fires (unanimity or $\theta = 0.75$, both splits). A: genuine computation; B: derived closed form evaluated; C: literal printed; D: misread computation. "Correct": programs whose executed answer equals the reference. "Rescue (grader)": the vote's answer differs from the reference only by a degree mark (grading artifact corrected before the tables were produced).

model	contest	reference / vote / override	outcome	A/B/C/D	correct	note
R1-8B						
	AIME 2025 q14	735 / 147 / 735	rescue	8/0/0/0	8/8	cube residues mod 3^7 enumerated
	AIME 2026 q28	157 / 12 / 157	rescue	8/0/0/0	8/8	compositions of 12 enumerated or by DP
	AIME 2026 q29	393 / 243 / 393	rescue	8/0/0/0	8/8	all 3^7 tuples checked
	CMIMC 2025 q8	2851 / 4193821 / 2851	rescue	8/0/0/0	8/8	trial-division factorization
	CMIMC 2025 q15	256 / 2 / 256	rescue	8/0/0/0	8/8	2^{16} grids or row-mask DP
	CMIMC 2025 q17	2240 / 1 / 0	wrong, harmless	0/0/8/0	0/8	no geometry; every program prints a guessed 0 or 1
	CMIMC 2025 q19	4102652 / 4102652 / 0	**regression**	0/1/7/0	0/8	six print a literal 0 after dead DP code
	CMIMC 2025 q33	44 / 85 / 44	rescue	8/0/0/0	5/8	permutations + shoelace; one init bug, two syntax errors
	HMMT Feb 2025 q13	2304 / 20 / 2304	rescue	8/0/0/0	7/8	self-avoiding-walk DFS; one BFS double-counts
	HMMT Feb 2026 q9	289 / 0 / 289	rescue	8/0/0/0	8/8	modular factorials, Lucas truncation faithful
	HMMT Feb 2026 q16	1574 / 2048 / 1048574	wrong, harmless	0/6/2/0	0/8	six evaluate $2^{20} - 2$ (separability ignored)
	SMT 2025 q12	79 / 64 / 79	rescue	7/0/0/1	6/8	512-configuration BFS; one targets the wrong cell
gpt-oss-20b						
	AIME 2025 q14	735 / 147 / 735	rescue	8/0/0/0	8/8	cube residues mod 3^7, convolution
	BRUMO 2025 q20	160 / 250 / 160	rescue	4/3/0/1	6/8	epistemic searches; three hard-code derived candidate sets
	BRUMO 2025 q24	$\frac{1+\sqrt{31}}{2}$ / 3.286 / $\sqrt{\frac{16+\sqrt{31}}{2}}$	rescue (equal form)	1/2/5/0	6/8	five print a hard-coded root; one solves the chord equation
	CMIMC 2025 q0	$\frac{13}{32}$ / "$\frac{13}{32}$ hours" / $\frac{13}{32}$	rescue	8/0/0/0	8/8	exact-fraction race simulation
	CMIMC 2025 q4	194832 / 126252 / 194832	rescue	8/0/0/0	8/8	multiples of 77 below 10^6 enumerated
	CMIMC 2025 q6	930 / 891 / 932	wrong, harmless (1 regr.)	7/0/1/0	0/8	seven faithful big-integer builds agree on 932 (see text)
	CMIMC 2025 q14	$\frac{138}{55}$ / $\frac{27}{11}$ / $\frac{138}{55}$	rescue	8/0/0/0	8/8	34,650 decks enumerated or exact recursion
	HMMT Feb 2026 q17	6992 / 2540 / 6992	rescue	7/0/0/1	7/8	61 lattice points enumerated; one uses radius 3
	HMMT Feb 2026 q26	74 / 74° / 74	rescue (grader)	7/0/0/1	7/8	coordinates and nsolve; one swaps a vector
	HMMT Nov 2025 q5	81920 / 20480 / 81920	rescue	7/1/0/0	8/8	DFS on the $4 \times 4 \times 4$ grid
	SMT 2025 q22	55 / 55° / 55	rescue (grader)	4/3/0/1	6/8	tangency geometry; three evaluate derived formulas
Qwen3-30B						
	AIME 2026 q29	393 / 243 / 393	rescue	8/0/0/0	8/8	brute force over 3^7 tuples
	CMIMC 2025 q6	930 / 891 / 932	wrong, harmless	8/0/0/0	0/8	exact big-integer sums give 932 (see text)
	CMIMC 2025 q33	44 / 64 / 44	rescue	6/0/2/0	6/8	hexagons by shoelace; two bare literals
	HMMT Feb 2025 q13	2304 / 146 / 2304	rescue	8/0/0/0	8/8	DFS on the 4×4 grid
	HMMT Feb 2026 q15	$2 - \pi/2$ / 21/16 / $69/16 - \pi$	wrong, harmless	0/1/7/0	0/8	seven print a sympy literal
	HMMT Feb 2026 q17	6992 / 8120 / 6992	rescue	8/0/0/0	7/8	61 lattice points; one mishandles ties
	SMT 2025 q38	$\frac{288\sqrt{527}}{527}$ / 25 / same	rescue	0/1/7/0	7/8	seven hard-code $288/527$; one evaluates the derived formula

wrong contested votes only 39 have the correct answer among the top-5 candidates; 138 of the 213 reference answers are symbolic expressions. The rules behave as on the main contests: no override worsens a question. Unanimity fires on one question ($+0.5$ points; 1/0 units), $\theta = 0.75$ on two ($+0.9$), and $\theta = 0.5$ on five ($+2.3$ [$+0.5$, $+4.7$]; 5/0), while programs alone gain $+3.8$ with 11 improved and 3 worsened units, and share-normalized CoT–PoT voting $+3.8$ with 13/5. Eight programs rarely agree unanimously on a symbolic answer, so the strict rules are nearly silent here and the looser threshold carries the gain; the ordering of precision against coverage is the one Table 3 predicts.

Table 31: Held-out ArXivMath 2026 (contested pools, one seed). Units improved / worsened; seed-averaged gain over the vote with a 95% bootstrap CI over questions; pool-level rescues / regressions; sign test over units.

rule	gpt-oss-20b	Δ [95% CI]	pools +/−	sign p
override $\theta = 1$, open	1/0	$+0.5$ [$+0.0$, $+1.4$]	$+1/-0$	1.000
override $\theta = 0.75$, open	2/0	$+0.9$ [$+0.0$, $+2.3$]	$+2/-0$	0.500
override $\theta = 0.5$, open	5/0	$+2.3$ [$+0.5$, $+4.7$]	$+5/-0$	0.062
override $\theta = 1$, candidates	1/0	$+0.5$ [$+0.0$, $+1.4$]	$+1/-0$	1.000
override $\theta = 0.5$, candidates	3/0	$+1.4$ [$+0.0$, $+3.3$]	$+3/-0$	0.250
programs alone (plurality)	11/3	$+3.8$ [$+0.5$, $+7.0$]	$+11/-3$	0.057
CPMaj, shares	13/5	$+3.8$ [$+0.0$, $+7.5$]	$+13/-5$	0.096

U INDEPENDENT PROGRAM BLOCKS PER SEED, AND MORE PROGRAMS

In the main tables a question's eight programs are shared by its seeds. Here gpt-oss-20b and Qwen3-30B receive a disjoint block of eight programs per seed (programs $8s$ to $8s+7$ for seed s; R1-8B was not resampled), so each seed is an independent replicate of the whole procedure. With independent blocks the pattern of Table 11 replicates on the two models: on test, the unanimous program override improves 5/0 units ($+0.8$ points) and the $\theta = 0.75$ override 7/1 units ($+1.8$); on development they improve 8/1 and 10/1 units. More units move than with shared programs, because a question can

now be rescued in one seed and not in another, while the seed-averaged gains stay close to the shared-program values for the same two models. Programs alone and share-normalized CoT–PoT voting remain two-sided (13/14 and 13/13 test units). Table 33 spends up to eight times the program budget. The $\theta = 0.75$ override does not gain from more programs ($+1.9$ on test with 64 programs, 6/0 units) but stays free of regressions; unanimity over 64 programs almost never holds (0/0 units); share-normalized CoT–PoT voting is the rule that benefits, reaching $+3.0$ on test with 64 programs (12/2 units), at eight times the rescue's cost and still with regressions, and programs alone gain $+2.4$ with 15/8 units.

Table 32: Independent program block per seed (gpt-oss-20b and Qwen3-30B, contested pools). Units improved / worsened per model; seed-averaged test gain over the vote with a 95% bootstrap CI over questions; pool-level rescues / regressions; development gain (units $+/-$).

rule	gpt-oss-20b	Qwen3-30B	test Δ [95% CI]	pools $+/-$	dev Δ (units)
programs alone (plurality)	9/2	4/12	$+0.5$ [-2.2, $+3.5$]	$+25/-21$	$+2.5$ (23/13)
CPMaj, shares	8/3	5/10	$+1.2$ [-1.4, $+4.1$]	$+26/-17$	$+3.0$ (24/8)
override $\theta = 0.5$, candidates	3/0	1/3	$+0.4$ [-0.5, $+1.5$]	$+6/-3$	$+2.5$ (15/1)
override $\theta = 1$, candidates	2/0	0/0	$+0.3$ [$+0.0$, $+0.7$]	$+2/-0$	$+1.0$ (7/0)
override $\theta = 0.5$, open	7/0	3/3	$+2.0$ [$+0.3$, $+4.2$]	$+18/-3$	$+2.6$ (16/4)
override $\theta = 0.75$, open	5/0	2/1	$+1.8$ [$+0.1$, $+3.8$]	$+14/-1$	$+1.9$ (10/1)
override $\theta = 1$, open	4/0	1/0	$+0.8$ [$+0.1$, $+1.8$]	$+6/-0$	$+1.2$ (8/1)

Table 33: More programs per contested question (gpt-oss-20b and Qwen3-30B; programs shared across seeds). Question-level gain over the vote (units $+/-$) with n programs.

	$n = 16$		$n = 32$		$n = 64$	
rule	test	dev	test	dev	test	dev
programs alone (plurality)	$+1.6$ (13/9)	$+2.7$ (20/8)	$+2.4$ (15/9)	$+1.9$ (17/9)	$+2.4$ (15/8)	$+2.1$ (18/9)
CPMaj, shares	$+2.6$ (13/5)	$+2.9$ (20/6)	$+2.7$ (12/4)	$+3.2$ (19/4)	$+3.0$ (12/2)	$+2.4$ (17/5)
override $\theta = 0.75$, open	$+1.5$ (5/0)	$+1.9$ (9/1)	$+1.1$ (5/1)	$+1.8$ (9/2)	$+1.9$ (6/0)	$+1.8$ (9/2)
override $\theta = 1$, open	$+0.1$ (1/0)	$+1.0$ (5/1)	$+0.0$ (0/0)	$+0.6$ (3/1)	$+0.0$ (0/0)	$+0.1$ (1/1)

V PROMPTS

All prompts of the rescue study and of the tournament are given verbatim; `<problem>` is the problem statement and `<candidates>` the candidate list, one per line as `(A) answer`, in an order shuffled per sample. Requests to gpt-oss-20b additionally carry the system message `Reasoning: high` (see the sampler paragraph of Appendix A for how its chat template renders it), extended for the program prompts by an instruction that no tools are available.

Chain of thought (pool).

```
Solve the following competition problem. Reason carefully and put your final answer in
\boxed{}.

Problem:
<problem>
```

Program (execution source).

```
Solve the following competition problem by writing a complete Python program that COMPUTES
the answer.

Rules (a response that violates them is invalid and will be discarded):
- Your final answer MUST be exactly one ```python code block containing a complete, runnable
  program. Stating the answer without such a program, or putting only `ANSWER: <value>` inside
  the block, is invalid.
- The program must compute the value by exhaustive search or enumeration, exact arithmetic
  with fractions or sympy, solving equations, or numerical evaluation with verification. A
  program whose only work is to print a number you derived in your head is invalid. If a
  direct search over the problem's actual parameters is infeasible, compute the answer from a
```

```
  formula that the program itself verifies on small cases.
- You cannot run code here (no tool or interpreter is available); never call a tool. Think
  first if you need to.
- Use only the Python standard library, sympy and numpy; it must finish within 60 seconds.
- The last line printed must be exactly 'ANSWER: <final answer>' with an exact value (an
  integer, a fraction like 7/3, or a simplified closed form like 3*sqrt(2)+1), not a decimal
  approximation unless the problem asks for one.

Problem:
<problem>
```

Hinted re-solve (same-model control).

```
Solve the following competition problem. Several independent solvers proposed the candidate
final answers listed below (in random order). At most one candidate is correct, and possibly
none is. Solve the problem yourself, and use the candidates to check your work: verify whether
each candidate satisfies the problem's conditions and identify the error behind any wrong
candidate. Finish with your own final answer in \boxed{} (it may be one of the candidates or
a different answer).

Problem:
<problem>

Candidate final answers:
<candidates>
```

Hinted program.

```
Several independent solvers proposed different final answers to the competition problem below
(listed in random order); at most one is correct, and possibly none is. Settle the
disagreement by writing a complete Python program that COMPUTES the answer.

Rules (a response that violates them is invalid and will be discarded):
- Your final answer MUST be exactly one ```python code block containing a complete, runnable
  program. Stating the answer without such a program, or putting only 'ANSWER: <value>' inside
  the block, is invalid.
- The program must compute the answer itself (exhaustive search, exact arithmetic with
  fractions or sympy, solving equations, or testing each candidate against every condition of
  the problem). A program whose only work is to print the candidate you believe is invalid.
  Where a candidate can be checked directly, check it.
- You cannot run code here (no tool or interpreter is available); never call a tool. Think
  first if you need to.
- Use only the Python standard library, sympy and numpy; it must finish within 60 seconds.
- The last line printed must be exactly 'ANSWER: <final answer>' with an exact value; it may
  differ from every candidate.

Problem:
<problem>

Candidate final answers:
<candidates>
```

Verification (same-model control; one prompt per candidate).

```
You are checking a proposed final answer to a competition problem.

Problem:
<problem>

Proposed final answer: <candidate>

Determine independently whether the proposed final answer is correct. Work the problem out
yourself; check the answer against every condition of the problem. End your response with
exactly one line of the form
VERDICT: CORRECT
or
VERDICT: INCORRECT
```

Tournament (Appendix C). The tournament seeds the top-8 answer clusters of a contested pool into a single-elimination bracket (1v8, 4v5, 3v6, 2v7). Each cluster is first condensed into a packet: the traces that reached that answer are summarized by the base model at temperature 0 (at most 2,500 tokens) with the prompt

```
Below are <k> independent solution traces that all conclude the answer is: <answer>

[Trace 1]:
```

```
<trace, truncated to 1,500 characters>
... (one block per trace)

Using the best evidence from ALL traces above, write a single structured summary in this
  format:

SUMMARY:
- Method: <one short phrase>
- Key steps:
  1. <first step with its numerical result>
  2. <second step with its numerical result>
  ... (continue as needed)
- Critical claims or assumptions:
  - <most important claim - state the actual value, not just a description>
  ... (list all critical ones)
- Main uncertainty or possible failure point:
  - <the step most likely to be wrong>

Important: Include ALL concrete intermediate values (numbers, probabilities, formulas)
  from any
  trace.
Do not omit numerical results. Do not add claims not supported by the traces.
Do not write a <think> block or private analysis. Start directly with the structured
  summary.
```

Each match is judged by the base model (or, in the ablations, Qwen3-8B) at $T{=}0.6$, top-p 0.95, with three votes per match (one vote in the quarter-finals of the in-run version; two votes per slot order in the symmetric-slot ablation); the majority of votes wins, and the bracket winner replaces the vote. The match prompt is

```
You are a careful verifier of candidate solutions.

Your job is to find which candidate is correct by checking their key claims, not by
  picking the
  most
polished or familiar-sounding answer.

Here is the problem:

<problem>

[Solution 1]: <answer A>
<packet summary A>

[Solution 2]: <answer B>
<packet summary B>

Verification procedure:

Step 1 - Find where the candidates disagree.
Look at the key numerical values, probabilities, or intermediate results that differ
  between the
candidates. These disagreement points are where errors hide.

Step 2 - For each disagreement point, independently verify which value is correct.
Work through the calculation yourself from first principles. Do NOT accept any candidate's
  claimed
value just because it is stated confidently or called a "standard result." A standard
  result
  may not
apply if the problem has constraints that violate the result's assumptions.

Step 3 - Eliminate the candidate whose key values you found to be wrong in Step 2.

Step 4 - If both remain, choose the one whose reasoning is most internally consistent and
  whose
intermediate values you verified.

Important:
- If the candidates cite different values for the same quantity, at most one can be right.
  Verify it
  yourself.
- A claim like "by symmetry," "it is well known," or "the standard probability is X" is
  not a
  proof.
  Check whether the stated conditions actually hold.

Which solution is correct, 1 or 2? Reply with just 1 or 2, then put the winning answer in
  \boxed{}
```

The vote is read from the boxed answer (falling back to the first "1" or "2"). Two stricter prompt variants (a completeness-check prompt and an adjudication prompt with an audit table) exist in the code and were not used for the numbers reported; the tournament code, its judge configurations (slot order, bracket permutation, vote schedule) and the replay scripts are in the supplementary material.

Multiple choice (GPQA). The chain-of-thought prompt asks for the letter of the final answer in `\boxed{}`; the hinted variant lists the proposed option letters and states that exactly one option is correct and that the proposals may all be wrong.

W SAMPLER, SANDBOX, HARDWARE AND TOKEN DETAILS

Sampler. All samples were generated with vLLM 0.13 on one GPU per job at $T{=}0.6$, top-p 0.95, one completion per request. gpt-oss-20b is served in its native MXFP4 weights with a 65,536-token context and at most 60,000 output tokens; Qwen3-30B-A3B-Thinking-2507 (FP8 release) and DeepSeek-R1-0528-Qwen3-8B use a 98,304-token context, at most 92,000 output tokens and an fp8 KV cache. The gpt-oss-20b requests carry the system message "Reasoning: high". The model's chat template places a request's system message among the developer instructions and writes its own default reasoning setting (medium) into the system header, where the harmony format specifies the reasoning level; the requests therefore ran at the template's default setting with "Reasoning: high" as an additional developer instruction, and we describe the model as run at its default setting; its per-sample accuracy on AIME 2025 is 75.8% with a median of 6.9k output tokens. All gpt-oss-20b samples of the study, including programs and controls, share this rendering. Every request carries a seed that is a deterministic function of (run seed, question, sample index, candidate index), and the candidate order in the hinted prompts is shuffled with the same seed, so an interrupted run resumes and reproduces exactly; samples are appended one JSON line at a time as they finish. Seeds of the study are disjoint blocks of 64 sample indices. 5.2% of gpt-oss-20b chain-of-thought samples reach the 60k budget and 1.6% of the finished ones carry no boxed answer (the final answer in display mathematics, or a stray tool call); both abstain from the vote.

Tool-integrated traces. For gpt-oss-20b, eight tool-integrated traces per contested question were sampled with the same sampler, seed scheme and system message, using the model's built-in Python tool in its harmony format: whenever a message ends with a call to `python`, the code runs in a persistent sandboxed session (notebook semantics, 60 s and 6 GB per call, the same interpreter and library restrictions as the programs) and its output is returned as a tool message before generation continues, for at most 24 calls per trace; the 60k output-token budget covers all turns. The answer is the last boxed expression of the final channel; truncated traces and traces without one abstain. Across the 1,104 traces, the mean is 12.5k output tokens and 15 tool calls, 38% of traces have at least one failed call, 11% at least one timed-out call, and 72% end with an answer.

Hardware. Sampling ran on single-GPU jobs with 48–141 GB of GPU memory that resume from their last written sample. The pools and evidence samples of the three models, the held-out and confirmatory contests and the additional program, tool-integrated and verification samples took about 1,080 GPU-hours (742 on preemptible jobs, 338 on reserved partitions for the last runs, most of them the confirmatory run); the tournament ablations and the additional R1-8B seeds and development pools took about 1,100 GPU-hours; the first R1-8B pools were sampled earlier on rented cloud GPUs.

Execution sandbox. Each program runs as `python -I` in an empty temporary directory with a 60 s wall-clock limit, a CPU-time limit, a 6 GB address-space limit, a 16 MB file-size limit and at most 64 processes; the process group is killed on timeout. Only the standard library, sympy and numpy are available. The evidence is the value on the last `ANSWER:` line of standard output. A program whose only statement prints a literal ("trivial") is discarded, as is a response without a complete code block; under the lenient first version of the prompt Qwen3-30B produced such responses for 29% of its samples, which is why the strict version above is the one used throughout.

Grading. Answers are compared by symbolic equivalence (sympy after normalization of the LaTeX form) and, for multiple choice, by option letter or option text.

Token counts. Table 34 gives mean output tokens. A contested question costs the pool 64 traces and the rescue eight programs; settled questions cost 16 traces and no programs.

Table 34: Output tokens. CoT trace and program: means per sample over the 306 questions (R1-8B CoT: tournament-experiment pools / new pools). Per contested question: eight programs and the 64-trace pool. Last column: programs as a share of all tokens of one run of the adaptive protocol (per seed; for R1-8B only its seed-0 pools on HMMT Nov 2025, SMT 2025 and CMIMC 2025, the pools sampled with token logs).

model	CoT trace	truncated	program	8 programs (contested q)	pool (contested q)	rescue share of a run
DeepSeek-R1-0528-Qwen3-8B	24.7k / 29.5k	0.2% / 0.1%	17.3k	151k	2.95M	4.8%
gpt-oss-20b	14.6k	5.2%	11.4k	135k	1.59M	7.6–7.8%
Qwen3-30B-A3B-Thinking	20.7k	0.1%	14.1k	174k	2.20M	5.3–5.7%