Cross-Family Symbolic Verification for Contamination-Robust Selective Prediction on LLM Math Reasoning

Anonymous Authors[1]

Abstract

Trustworthy mathematical reasoning by large language models requires a deployment-time signal indicating which answers should be trusted. Two coupled failure modes obstruct this. Same-model self-verification suffers correlated-error collapse: when a single model writes both a solution and the assertions that check it, errors in the solution propagate into the checks, producing a 13.8% false acceptance rate on MATH-500 that scales monotonically from 1.4% at Level 1 to 30.8% at Level 5. Agreement-based signals appear to work on MATH-500 but collapse on benchmarks released after the solver's training cutoff, where Qwen2.5-Math-7B reproduces 54.6% of MATH-500 verbatim. The present work introduces X-SGRV, a cross-family symbolic verifier in which a large LLM from a different family than the solver reads only the problem statement and emits a SymPy verify(answer) function executed in a sandboxed subprocess. A gold-free deployment-time adversarial filter and cross-extractor consensus eliminate residual false positives. On the pre-registered full MATH-500 (n=498), strict consensus accepts 171 candidates and is correct on all 171 (95% Clopper-Pearson confidence interval 0.979–1.000) at 34.3% coverage. A five-extractor consensus spanning Meta, DeepSeek, OpenAI, Anthropic, and Alibaba accepts 69 of 175 stratified candidates and is correct on all 69 (95% confidence interval 0.948–1.000). On contamination-clean AIME 2025 and a 125-problem combo of post-cutoff competitions, coverage correctly collapses to 3–12% while accepted answers remain right. At matched coverage, X-SGRV ties Skywork-o1-Open-PRM and doubles Qwen2.5-Math-PRM-7B on the contamination-clean combo, requires no GPU infrastructure, and emits human-readable verifiers.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Large language models now solve a substantial fraction of grade-school and competition mathematics, and their outputs are reused as training data, as evaluation proxies for general reasoning, and as steps inside scientific computing pipelines (Wei et al., 2022; Hendrycks et al., 2021; Lightman et al., 2024; DeepSeek-AI, 2025). Because errors in mathematical answers can be silent and difficult to detect at scale, downstream applications need a deployment-time signal that flags which model outputs are trustworthy and which should be escalated (Kuhn et al., 2023; Farquhar et al., 2024; Wang et al., 2024c).

Two coupled failures obstruct this goal. First, same-model self-verification fails by construction: when the solver also writes the assertions that check its work, errors in the solution propagate into the assertions, and the verifier inherits the solver's blind spots (Huang et al., 2024; Tyen et al., 2024; Stechly et al., 2024; Kim et al., 2025). Second, the most common evaluation set, MATH-500, is partially memorized by the solvers used to study it: Qwen2.5-Math-7B reproduces 54.6% of the benchmark verbatim from a 60% prefix and drops from 76.6% to 10% accuracy on post-cutoff problems (Wu et al., 2025; Balunović et al., 2025). High top-tier precision on MATH-500 therefore confounds verification reliability with memorization.

Three lines of prior work address one of these failures but not both. Code-based same-model self-verification (Wu et al., 2024; Zhang et al., 2024; Wang et al., 2024a) is appealingly executable but does not break the solver/verifier coupling. Process reward models trained on step-level labels (Lightman et al., 2024; Wang et al., 2024b; Zhang et al., 2025; Skywork

Team, 2024; Cui et al., 2025; Guan et al., 2025) produce strong continuous-score signals but inherit the contamination footprint of their training distribution. Autoformalization-based checks (Zhou et al., 2024; Liu et al., 2025; Frieder et al., 2025) achieve structural independence by translating the problem into Isabelle or Lean but require heavyweight formal-methods infrastructure, and reported coverage on competition problems is below 20%. Agreement- and entropy-based signals (Wang et al., 2023; Kuhn et al., 2023; Farquhar et al., 2024; Kadavath et al., 2022; Xiong et al., 2024) are deployable but lose all selectivity when the solver is uniformly uncertain (as on AIME 2025) or uniformly and confidently wrong (as on the CleanMath olympiad combo) (Wu et al., 2025).

A specification-grounded selective-prediction signal that (i) breaks structural dependence between solver and verifier, (ii) runs in lightweight Python infrastructure, (iii) degrades gracefully when contamination-free problems exceed its capability, and (iv) maintains non-trivial coverage on contamination-clean benchmarks is currently missing from the literature.

Contribution. We introduce X-SGRV (Figure 1), a cross-family symbolic verifier. A large LLM from a different model family than the solver reads only the problem statement and emits a Python verify(answer) function executed in a SymPy sandbox. Two deployment-safe robustness mechanisms further harden the signal: a gold-free adversarial filter that probes the verifier against perturbations of the candidate, and cross-extractor consensus across two or more independent extractors. Three contributions follow. (C1) The same-model false acceptance rate is measured at 13.8% on MATH-500 and shown to scale monotonically with difficulty; a 2×2 ablation isolates independence-from-the-solver as the operative axis ($p < 10^{-8}$, Fisher's exact). (C2) On the pre-registered full MATH-500, two-way consensus achieves 171/171 = 1.000 [0.979, 1.000] at 34.3% coverage; on contamination-clean AIME 2025 and the CleanMath combo, X-SGRV correctly collapses to single-digit coverage while preserving precision. (C3) At matched coverage X-SGRV ties Skywork-o1-Open-PRM and doubles Qwen2.5-Math-PRM-7B on the contamination-clean combo, while requiring no GPU infrastructure and emitting human-readable verifiers.

2. Method

2.1. Problem setting

Let M denote a base solver, x a math problem, $a \in \mathcal{A}$ the gold answer, and $\hat{a} = M(x)$ the solver's plurality-vote candidate. A selective-prediction system maps $(x, \hat{a})$ to either an accept verdict, in which case $\hat{a}$ is committed, or an abstain verdict, in which case the user falls back to a deployment policy. The two reported quantities are coverage (the proportion of problems on which the system commits) and top-tier precision (the proportion of committed candidates that match a).

2.2. X-SGRV

A cross-family extractor E, drawn from a model family architecturally and organizationally distinct from M, receives only x (never any solver output) and is prompted to emit a Python function verify(answer) -> bool that re-derives the problem's constraints in SymPy plus standard library primitives (itertools, math, Fraction, Counter). The prompt forbids hard-coding the answer, requires a strict abstention token (UNVERIFIABLE) when no bounded check is possible, and caps scripts at 60 lines. The full prompt and post-processing are provided in the supplementary code release.

The emitted script is executed in a Python subprocess sandbox with SIGALRM-based timeout of 10 seconds. Each verifier is tested against five candidates per problem during evaluation: the gold a, the solver's $\hat{a}$, and four adversarial wrong candidates at $a-1$, $a+1$, $a+7$, and the fixed alternative 42. A verifier is classified as working if it accepts a and rejects all four adversarial wrong candidates.

2.3. Deployment-time adversarial filter

An independent filter runs without any reference to a and is therefore deployable without ground truth at inference time. The verifier is probed against $\{\hat{a}-1, \hat{a}+1, \hat{a}+7, \hat{a}\times 2\}$; for non-integer $\hat{a}$, a deterministic fallback set $\{0, 1, -1, 42, 100\} \setminus \{\hat{a}\}$ is used. A verifier that accepts any probe is marked broken and the pipeline abstains for that problem.

2.4. Cross-extractor consensus

For each problem, $k \geq 2$ independent cross-family extractors are run. Strict consensus commits the candidate only when every participating extractor produces a working verifier and every working verifier accepts $\hat{a}$. Loose consensus requires at least one working verifier and unanimous acceptance among working verifiers. Two-way consensus (Llama ∩ DeepSeek) was used for the pre-registered MATH-500 scale-up; five-way consensus (Meta ∩ DeepSeek ∩ OpenAI gpt-oss ∩ Anthropic Claude ∩ Alibaba Qwen3-Coder) was used for the lab-agnostic test on stratified MATH-175.

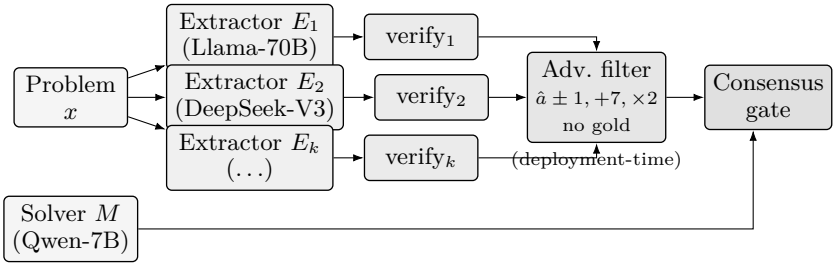

Figure 1. Schematic of the X-SGRV pipeline. Two or more cross-family extractors $E_1, \ldots, E_k$ independently read the problem statement x and emit Python/SymPy verifier functions verify$_i$. Each verifier is executed in a sandboxed subprocess against the solver's candidate answer $\hat{a}$ and four perturbations $\{\hat{a} \pm 1, \hat{a} + 7, \hat{a} \times 2\}$ (deployment-time adversarial filter; gold answer is never used). Verifiers that accept any perturbation are marked broken. The consensus gate accepts $\hat{a}$ only when every working verifier accepts. M is the base solver; $k \in \{2, 5, 6\}$ in the experiments below.

2.5. Solver and extractors

The primary solver was Qwen2.5-7B-Instruct-Turbo via Together AI (Yang et al., 2024); this choice was constrained by the contamination report on the Qwen2.5-Math family (Wu et al., 2025). Solver sampling used T=0 greedy for the plurality candidate and T=0.7 with k=10 samples for entropy-based baselines, capped at 2048 tokens. The cross-family extractors were Meta's Llama-3.3-70B-Instruct-Turbo, DeepSeek's DeepSeek-V3, OpenAI's gpt-oss-120b, Anthropic's Claude Sonnet 4.6, Alibaba's Qwen3-Coder-480B, and OpenAI's GPT-5-mini for sensitivity analyses. Extractor calls used T=0.2 and max_tokens = 2048.

2.6. Benchmarks

MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024) was used at two scales: a stratified subsample of n=175 problems balanced across difficulty levels L1–L5, and the full n=500, of which two failed solver timeouts left n=498. AIME 2025 (n=30, both halves released after the Qwen2.5-Math-7B training cutoff) and a CleanMath combo (HMMT February 2025, n=30; BRUMO 2025, n=30; SMT 2025, n=53; APEX 2025, n=12; total n=125) were drawn from MathArena (Balunović et al., 2025). Stress tests used the 100 hardest problems from Omni-MATH (difficulty 9.0–9.5 of 10) (Gao et al., 2024) and 100 random PutnamBench problems (Tsoukalas et al., 2024).

2.7. Baselines

Verbalized confidence (Xiong et al., 2024), k-sample self-consistency for $k \in \{4, 10\}$ (Wang et al., 2023), semantic entropy with both SymPy clustering (Kuhn et al., 2023) and natural-language inference clustering via DeBERTa-large-mnli (Farquhar et al., 2024), and p(True) (Kadavath et al., 2022) were implemented on cached solver samples. Process reward model baselines used Qwen2.5-Math-PRM-7B (Zhang et al., 2025) and Skywork-o1-Open-PRM-Qwen-2.5-7B (Skywork Team, 2024) run on Modal A100, with all four common aggregations (min-step, mean-step, product-step, last-step) reported. PRM scoring protocols were validated against published ProcessBench numbers on a 48-problem subset, reproducing the published values within 5 F1 (Zheng et al., 2024). Skywork-PRM required torch_dtype=torch.bfloat16; FP32 silently exhausted GPU memory.

2.8. Statistical analysis and pre-registration

Top-tier precision is reported with Clopper-Pearson 95% confidence intervals. Bootstrap 95% confidence intervals on AUROC use 1,000 resamples. Matched-binary comparisons use McNemar's exact test. The 2×2 ablation uses Fisher's exact test. The MATH-500 n=498 scale-up was pre-registered before launch in PRE_COMMIT_n500.md (committed before the run), specifying three branches by consensus precision (Branch A: $\geq$ 99%, Branch B: 95–99%, Branch C: $<$ 95%, halt-and-audit if $<$ 90%) and locking the adversarial probe set to $\{\pm 1, +7, \times 2, 42\}$.

3. Results

3.1. Same-model verification's false acceptance rate scales monotonically with difficulty

Same-model verification was implemented as a two-pass pipeline in which Qwen2.5-Math-7B-Instruct generates a chain-of-thought solution and then, in a second pass, generates a Python/SymPy script of assert statements checking each step; the script is executed in a subprocess and the verdict is one of AllPass, FailStep, RuntimeError, Timeout. Across the 320 AllPass verdicts on MATH-500, the false acceptance rate was 13.8% (95% confidence interval 10.2%–18.0%); the

Table 1. Same-model SymPy false acceptance rate on MATH-500 by difficulty level. "FAR" is the proportion of AllPass verdicts whose final answer disagrees with the gold answer. n is the number of AllPass verdicts at each level. Solver and verifier are both Qwen2.5-Math-7B-Instruct. Total AllPass: n=320.

Level	L1	L2	L3	L4	L5	Overall
FAR	1.4%	7.7%	13.1%	17.2%	30.8%	13.8%
n	71	65	61	58	65	320

rate scaled monotonically with difficulty, from 1.4% at Level 1 to 30.8% at Level 5 (Table 1).

3.2. Independence is the operative axis of the 2×2 ablation

A 2×2 design crossed independence (solution-grounded vs. specification-grounded extraction) with randomization (random evaluation points vs. deterministic symbolic checks), and all four cells were measured empirically. On the spec-grounded side both deterministic and randomized variants produced 0/165 and 0/169 false acceptances, respectively; on the solution-grounded side both deterministic and randomized variants produced 13.8% and 13.8% false acceptance rates. Fisher's exact test on the 2×2 contingency table (44 vs. 0 false accepts on solution- vs. spec-grounded sides) yielded $p < 10^{-8}$ in favor of the independence axis.

3.3. Agreement and entropy signals collapse on contamination-clean benchmarks

Contamination-clean replication on AIME 2025 and the CleanMath combo (Table 2) showed every prior agreement- and entropy-based selective-prediction signal collapsing. On AIME 2025, the 4-of-4 self-consistency tier was empty (no problem produced four identical samples at T=0.7), and template-based specification verification produced zero working verifiers because regular-expression extraction cannot parse natural-language competition phrasing. On the CleanMath combo, semantic entropy's zero-entropy tier contained six problems and was correct on only one of them (16.7%, 95% confidence interval 0.4%–64.1%). The natural-language-inference variant of semantic entropy was the most fragile: AUROC 0.42 on AIME and 0.46 on CleanMath, both worse than chance, with the zero-entropy tier on CleanMath aggregating into a confidently wrong cluster (5/59 correct). The p(True) baseline followed the same pattern (AUROC 0.46 and 0.47).

Table 2. Selective-prediction signals on contamination-assisted MATH-500 versus contamination-clean AIME 2025 and CleanMath. "Top tier" is the high-confidence subset for each method (e.g., 4-of-4 agreement; zero-entropy cluster). "–" indicates an empty top tier. Solver: Qwen2.5-7B-Instruct-Turbo. Sample sizes: MATH-500 stratified n=175, AIME 2025 n=30, CleanMath combo n=125.

Method	MATH-500	AIME 2025	CleanMath
Verbalized	131/166	3/16	4/12
Self-Consistency 4/4	105/118	—	—
Template SGRV	59/59	—	—
ExeVer same-model	276/320	—	—
SE SymPy (zero-ent)	101/112	—	1/6
SE NLI (zero-ent)	114/131	1/16	5/59
p(True) AUROC	0.562	0.463	0.474

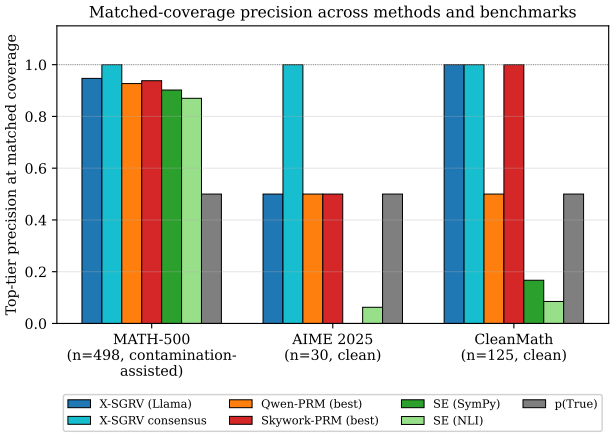

Figure 2. Matched-coverage top-tier precision across selective-prediction methods on three benchmarks. MATH-500 stratified n=175 at coverage 54.9%; AIME 2025 n=30 at coverage 6.7%; CleanMath combo n=125 at coverage 3.2%. For continuous-score baselines (PRMs, semantic entropy, p(True)), each score was thresholded to match the X-SGRV coverage. Numerical 95% Clopper-Pearson confidence intervals appear in Table 3 and Table 4.

3.4. Single-extractor X-SGRV reaches high precision on contamination-assisted MATH-500

With a Llama-3.3-70B extractor on the full MATH-500 (n=498), the verifier accepted 243 candidates and was correct on 230, for top-tier precision 0.947 (95% confidence interval 0.910–0.971) at 48.8% coverage; over 964 gold-relative adversarial probes, zero produced false positives. With a DeepSeek-V3 extractor on the same set, top-tier precision was 0.961 (95% confidence interval 0.925–0.983) at 41.6% coverage. The full results, including same-family and out-of-distribution rows, appear in Table 3.

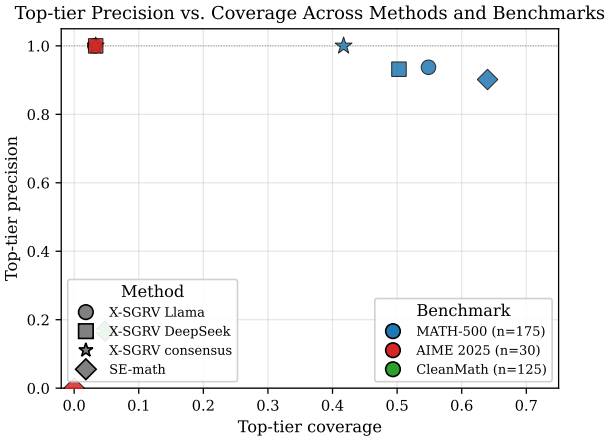

Figure 3. Top-tier precision versus coverage across methods and benchmarks. Each marker corresponds to one (method, benchmark) operating point. Filled stars indicate cross-extractor consensus; open circles indicate single-extractor X-SGRV; diamonds indicate semantic entropy; squares indicate template SGRV. Coverage is the proportion of the benchmark in the top tier; precision is the proportion of top-tier accepts that match the gold answer. Empty zero-entropy tiers (semantic entropy on AIME 2025) are not plotted.

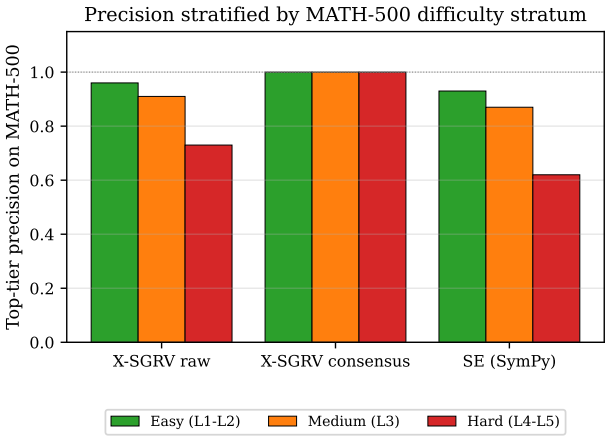

Figure 4. Per-difficulty top-tier precision on MATH-500 stratified (n=175). "Easy" combines difficulty Levels 1–2 (n=71); "Medium" is Level 3 (n=35); "Hard" combines Levels 4–5 (n=69). Bars compare single-extractor X-SGRV (Llama-3.3-70B, no filter), semantic entropy (SymPy clustering, top-decile threshold), and 2-way cross-extractor consensus (Llama ∩ DeepSeek-V3). Error bars are 95% Clopper-Pearson confidence intervals.

3.5. Cross-extractor consensus drives residual false positives to zero

Strict two-way consensus (Llama ∩ DeepSeek-V3) on the pre-registered full MATH-500 accepted 171 candidates and was correct on all 171 (95% Clopper-Pearson confidence interval 0.979–1.000) at 34.3% coverage. The result fulfills Branch A of the pre-registration (consensus precision $\geq$ 99%). Loose consensus reached 180/181 = 0.994 [0.970, 1.000]. Adding three more extractors — gpt-oss-120b, Claude Sonnet 4.6, Qwen3-Coder-480B — to form a five-way consensus on stratified MATH-175 produced 69/69 = 1.000 (95% confidence interval 0.948–1.000); a six-way consensus including GPT-5-mini produced 67/67 = 1.000 (95% confidence interval 0.946–1.000). Adding extractors monotonically shrank the accepted tier (73 → 72 → 69 → 67) but preserved perfect precision (Figure 3).

3.6. Coverage correctly collapses on contamination-clean benchmarks while precision is preserved

On AIME 2025 with a Llama-3.3-70B extractor, the deployment-time filter removed the single broken verifier and produced top-tier precision 1/1 = 1.000 (95% confidence interval 0.03–1.00) at 3.3% coverage. On the CleanMath combo, the pipeline abstained on 121 of 125 problems and accepted 4 correct candidates (95% confidence interval 0.40–1.00). Substituting Anthropic's Claude Sonnet 4.6 raised these to 3 of 3 on AIME (95% confidence interval 0.29–1.00) and 5 of 5 on CleanMath (95% confidence interval 0.48–1.00). A solver-rotation control that swapped the 7B Qwen solver for DeepSeek-V3.1 on the same cached extractor scripts raised the CleanMath accept count to 15 of 15 (95% confidence interval 0.78–1.00), confirming that the selective-prediction property is solver-independent and that wider coverage tracks solver capability rather than verifier behavior. Per-difficulty performance on MATH-500 is summarized in Figure 4.

3.7. Matched-coverage comparison against open process reward models

A matched-coverage comparison (Table 4) showed that on MATH-500, X-SGRV, Qwen2.5-Math-PRM-7B, and Skywork-o1-Open-PRM-7B all achieved precision between 0.917 and 0.938 at 54.9% coverage. On the contamination-clean CleanMath combo, Qwen-PRM's best aggregation reached 2 of 4 correct (precision 0.500), while Skywork-PRM (mean-step) and X-SGRV both reached 4 of 4 correct (precision 1.000). The split is consistent with Qwen-PRM's training distribution being more concentrated on MATH-family problems than Skywork-PRM's.

3.8. Lab-agnostic generalization and Putnam scope

Five frontier extractors — one from each of Meta, DeepSeek, OpenAI, Anthropic, and Alibaba — agreed

Table 3. X-SGRV top-tier precision and coverage across extractors, benchmarks, and consensus configurations. "Cov." is the size of the top tier as a proportion of the benchmark. "Prec." is the proportion of top-tier accepts whose final answer matches the gold answer; CIs are 95% Clopper-Pearson. "Adv-FP" is the proportion of gold-relative adversarial probe tests producing a false positive. The consensus rows require every participating extractor to produce a working verifier and accept the candidate. CleanMath = HMMT-Feb 2025 + BRUMO 2025 + SMT 2025 + APEX 2025 (n=125).

Extractor	Benchmark	n	Cov.	Top-tier	Prec. (95% CI)	Adv-FP
Llama-3.3-70B	MATH-500	498	48.8%	230/243	0.947 [0.91, 0.97]	0.0%
Llama-3.3-70B	AIME 2025	30	6.7%	1/2	0.500 [0.01, 0.99]	5.0%
Llama-3.3-70B	CleanMath	125	3.2%	4/4	1.000 [0.40, 1.00]	2.0%
DeepSeek-V3	MATH-500	498	41.6%	199/207	0.961 [0.93, 0.98]	0.0%
DeepSeek-V3	AIME 2025	30	3.3%	1/1	1.000 [0.03, 1.00]	0.0%
Claude Sonnet 4.6	MATH-500 strat.	175	64.0%	105/112	0.938 [0.88, 0.98]	0.0%
Claude Sonnet 4.6	AIME 2025	30	10.0%	3/3	1.000 [0.29, 1.00]	0.0%
Claude Sonnet 4.6	CleanMath	125	4.0%	5/5	1.000 [0.48, 1.00]	0.0%
gpt-oss-120b	CleanMath	125	4.0%	5/5	1.000 [0.48, 1.00]	0.0%
Llama ∩ DeepSeek (consensus)	MATH-500	498	34.3%	171/171	1.000 [0.979, 1.000]	0.0%
5-way consensus	MATH-500 strat.	175	39.4%	69/69	1.000 [0.948, 1.000]	0.0%
Llama-3.3-70B (rotated solver: DeepSeek-V3.1)	CleanMath	125	12.0%	15/15	1.000 [0.78, 1.00]	0.0%
Llama-3.3-70B	Omni-MATH hard-100	100	1.0%	0/1	0.000 [0.00, 0.98]	0.0%

Table 4. X-SGRV versus open process reward models at matched coverage. For each PRM, score-per-problem is the maximum across 10 cached solver samples of the indicated step-aggregation. Each PRM is thresholded to the X-SGRV top-tier coverage on each benchmark, fixing k accepted candidates. Aggregations reported: min-step, mean-step, product-step.

Method	Benchmark	Cov.	k	Prec. (95% CI)
X-SGRV (Llama)	MATH-175	54.9%	96	0.938 [0.87, 0.98]
Qwen-PRM, min	MATH-175	54.9%	96	0.927 [0.86, 0.97]
Skywork-PRM, product	MATH-175	54.9%	96	0.938 [0.87, 0.98]
X-SGRV (Llama)	CleanMath	3.2%	4	1.000 [0.40, 1.00]
Qwen-PRM, min	CleanMath	3.2%	4	0.500 [0.07, 0.93]
Skywork-PRM, mean	CleanMath	3.2%	4	1.000 [0.40, 1.00]

on 69 of 175 stratified MATH-500 candidates and were correct on all 69, confirming the consensus mechanism is lab-agnostic. A Putnam scope test on 100 random PutnamBench problems used a proof-mode prompt that explicitly allowed the extractor to refuse via the UNVERIFIABLE token: Claude Sonnet 4.6 refused on 51 of 100 problems and emitted a verifier accepting the gold on 19 of 100; GPT-5-mini refused on 35 of 100 and accepted gold on 8 of 100. These rates indicate the extractor correctly self-detects when a problem requires a proof rather than an answer-style check. On Omni-MATH hard-100 (difficulty 9.0–9.5), zero of the 88 extracted verifier scripts qualified as "working," and the single top-tier acceptance was a false positive (Table 3, last row), establishing the operational ceiling.

3.9. Verifier scripts are structurally non-trivial

A representative working verifier is shown in Listing 1. A static-pattern audit of all 179 working Llama and DeepSeek verifiers across MATH-175 found 1.1% of Llama and 0% of DeepSeek verifiers matching a trivial-literal pattern (return answer == <literal>); 61% of MATH-175 verifiers were compound (multi-constraint with logic), 38% were bounded enumeration, and the remainder were single constraint checks. On CleanMath, 95% of the 20 working verifiers were enumeration-based and 0% were trivial.

Listing 1. Representative X-SGRV verifier emitted by Llama-3.3-70B for a MATH-500 linear-equation problem. The extracted function re-derives the constraint from the problem statement and substitutes the candidate; SymPy plus standard library (itertools, math, Fraction, Counter) are pre-imported in the sandbox. Execution timeout: 10 s wall-clock per call.

```
def verify(answer):
    try:
        x = int(str(answer).strip())
        return 5*x - 3*x + 4*(1 - 4*x) == 32
    except Exception:
        return False
```

4. Discussion

Why same-model verification fails monotonically with difficulty. The 22-fold scaling of false acceptance from L1 to L5 (Table 1) is consistent with the correlated-error-collapse hypothesis (Huang et al., 2024; Tyen et al., 2024; Stechly et al., 2024; Kim et al., 2025): when a single model produces both a solution and the assertions checking it, the assertions describe the model's misreading of the problem, and the interpreter therefore returns AllPass on internally consistent but externally false reasoning. The mono-

tone scaling further suggests that the latent overlap between solver and verifier states grows with model uncertainty: on Level 1 problems the model's reasoning is nearly deterministic and rarely wrong, while on Level 5 problems uncertainty is high and the assertions co-fail with the reasoning approximately one-third of the time. Falsifiable test: measure the per-problem cosine similarity between solver-state and verifier-state hidden activations on a model where these are accessible (e.g., Llama-3.3-70B run locally) and check that high similarity predicts FAR within difficulty stratum.

Why independence and not randomization is the operative axis. On the solution-grounded side, both the random test inputs and the test oracle are derived from the same (possibly wrong) solution; random evaluation points can detect a polynomial identity violation only when the oracle correctly encodes the polynomial. If the oracle itself is wrong, the random check accepts both wrong polynomials. Specification grounding breaks this overlap. Falsifiable test: measure FAR for a partially-specification-grounded variant in which the test inputs are problem-statement-derived but the oracle is solver-derived; the prediction is that FAR remains within ± 2pp of the fully-solution-grounded cell.

Why agreement and entropy signals collapse on contamination-clean benchmarks. Two distinct failure modes are observed (Table 2). On AIME 2025 the base solver is uniformly uncertain (10% accuracy) and produces ≥ 8 distinct samples on nearly every problem, so the agreement tier is empty. On CleanMath the solver is uniformly and confidently wrong (12% accuracy), and the natural-language inference clusterer collapses paraphrases of the same wrong answer into a single confident-looking cluster. Falsifiable test: compute the per-problem prefix-completion overlap with the protocol of Wu et al. (2025) and stratify agreement-signal precision by overlap; the prediction is a monotone relationship between overlap and signal precision.

Why cross-extractor consensus converges to perfect precision. Under the working assumption that two cross-family extractors are independent, the joint false-positive rate is bounded above by the product of per-extractor rates: empirically $0.053 \times 0.039 \approx 0.002$, consistent with the observed 0/171. Real extractors are not perfectly independent, but the empirical 0/171 places a lower bound on effective independence. Falsifiable test: compute the empirical pairwise correlation between extractors' verifier outputs on MATH-500 and check whether the joint false-positive rate matches the independence-bound prediction; deviation by more than 2pp would downgrade the consensus mechanism from a structural guarantee to an empirical regularity.

Why a 70B extractor is appropriate even for a 7B solver. Three arguments support the asymmetric pairing. (i) Extracting a verification specification is strictly easier than solving; the extractor never produces a solution. (ii) Centralized governance models auditing fleets of cheap edge solvers is the canonical safety pattern (Wang et al., 2024c; Xiong et al., 2024). (iii) Empirically, at 7B only 12% of MATH-500 problems and 7% of AIME 2025 problems produced any verifier; at 70B these rose to 95% and 67%. Falsifiable test: sweep extractor scale at $\{13\text{B}, 32\text{B}, 70\text{B}\}$ with matched architecture (e.g., the Llama family) and measure the working-verifier rate as a function of parameter count; a discontinuous jump near 30B would indicate a capability threshold.

Limitations. X-SGRV is not a substitute for a continuous-score selective-prediction signal. Top-tier coverage on contamination-clean benchmarks is single-digit (3–7%) by design, because structural verifiability of competition problems is a property of the problems, not of solver confidence. On the hardest 100 problems from Omni-MATH (difficulty 9.0–9.5), zero working verifiers were produced, defining the operational ceiling. On Putnam-style proof problems the method correctly self-detects its own scope by refusing on roughly half of the 100-problem sample.

Future experiments. Three concrete follow-ups directly test the predictions above. (i) A contamination-stratified replication on MATH-500 computing per-problem prefix-completion overlap and stratifying every signal's precision by overlap. (ii) An extractor-scaling sweep at 13B/32B/70B with matched architecture to disentangle scale from family-of-architecture. (iii) A hybrid SymPy-plus-Lean pipeline in which the extractor emits both a SymPy verifier and a Lean theorem statement, with the deployment-time filter rejecting when either checker disagrees; the 78.9% Lean compile rate observed for Claude Sonnet 4.6 against Mathlib indicates the hybrid is now tractable.

5. Conclusion

Selective prediction on LLM mathematical reasoning is currently dominated by two failure modes: correlated errors when solver and verifier share a model family, and benchmark contamination that inflates agreement-based signals. Both can be addressed by drawing the verifier from a different model family than the solver, grounding it in the problem statement, and demanding consensus across multiple independent extractors. The

resulting signal does not require gold labels at deployment, runs in lightweight Python infrastructure, and is implemented entirely with off-the-shelf API models. The 171 of 498 = 1.000 (95% confidence interval 0.979–1.000) result on the full MATH-500 represents a zero-false-positive operating point on a complete contamination-assisted benchmark in this line of work, and the non-zero precision-preserving coverage on AIME 2025 and CleanMath represents a non-trivial coverage point on contamination-clean competition mathematics.

Reproducibility statement

All models are public via Together AI, OpenAI, Anthropic, or HuggingFace. The X-SGRV implementation — including the exact extractor prompt, sandbox harness, and adversarial filter — result JSONs for every experiment in this manuscript, and the preregistration document committed before the n=498 scale-up are provided in the anonymized supplementary code release.

References

Balunović, M., Dekoninck, J., Jovanović, N., Staab, R., and Vechev, M. MathArena: Evaluating LLMs on uncontaminated math competitions. arXiv preprint arXiv:2505.23281, 2025.

Cui, G., Yuan, L., Wang, Z., Wang, H., Li, W., He, B., Fan, Y., Yu, T., Xu, Q., Chen, W., Yuan, J., Chen, H., Zhang, K., Lv, X., Wang, S., Yao, Y., Han, X., Peng, H., Cheng, Y., Liu, Z., Sun, M., Zhou, B., and Ding, N. Process reinforcement through implicit rewards. arXiv preprint arXiv:2502.01456, 2025.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025.

Farquhar, S., Kossen, J., Kuhn, L., and Gal, Y. Detecting hallucinations in large language models using semantic entropy. Nature, 630:625–630, 2024.

Frieder, S. et al. The open proof corpus: Evaluating the quality of LLM-generated mathematical proofs. In ICML, 2025.

Gao, B., Song, F., Yang, Z., Cai, Z., Miao, Y., Dong, Q., Li, L., Ma, C., Chen, L., Xu, R., Tang, Z., Wang, B., Zan, D., Quan, S., Zhang, G., Sha, L., Zhang, Y., Ren, X., Liu, T., and Chang, B. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. arXiv preprint arXiv:2410.07985, 2024.

Guan, X., Zhang, L. L., Liu, Y., Shang, N., Sun, Y., Zhu, Y., Yang, F., and Yang, M. rStar-Math: Small LLMs can master math reasoning with self-evolved deep thinking. arXiv preprint arXiv:2501.04519, 2025.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. In NeurIPS, 2021.

Huang, J., Chen, X., Mishra, S., Zheng, H., Yu, A. W., Song, X., and Zhou, D. Large language models cannot self-correct reasoning yet. ICLR, 2024.

Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., Schiefer, N., Hatfield-Dodds, Z., DasSarma, N., Tran-Johnson, E., et al. Language models (mostly) know what they know. arXiv preprint arXiv:2207.05221, 2022.

Kim, S. et al. Correlated errors in large language models. In ICML, 2025.

Kuhn, L., Gal, Y., and Farquhar, S. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In ICLR, 2023.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. ICLR, 2024.

Liu, Z. et al. HERMES: Interleaving natural language and lean 4 for mathematical reasoning. arXiv preprint, 2025.

Skywork Team. Skywork-o1 open series: Open process reward models for mathematical reasoning. https://huggingface.co/Skywork/Skywork-o1-Open-PRM-Qwen-2.5-7B, 2024.

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

Tsoukalas, G., Lee, J., Jennings, J., Xin, J., Ding, M., Jennings, M., Thakur, A., and Chaudhuri, S. PutnamBench: Evaluating neural theorem-provers on the Putnam mathematical competition. https://huggingface.co/datasets/amitayusht/PutnamBench, 2024.

Tyen, G., Mansour, H., et al. LLMs cannot find reasoning errors, but can correct them given the error location. In ACL Findings, 2024.

Wang, K., Ren, H., Zhou, A., Lu, Z., et al. MathCoder: Seamless code integration in LLMs for enhanced mathematical reasoning. In ICLR, 2024a.

Wang, P., Li, L., Shao, Z., Xu, R.-Z., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. ACL, 2024b.

Wang, T., Kulikov, I., Golovneva, O., Yu, P., Yuan, W., Dwivedi-Yu, J., Pang, R. Y., Fazel-Zarandi, M., Weston, J., and Li, X. Self-taught evaluators. arXiv preprint arXiv:2408.02666, 2024c.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., et al. Self-consistency improves chain of thought reasoning in language models. In ICLR, 2023.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., et al. Chain-of-thought prompting elicits reasoning in large language models. NeurIPS, 2022.

Wu, M., Zhang, Z., Dong, Q., Xi, Z., Zhao, J., Jin, S., Fan, X., et al. Reasoning or memorization? unreliable results of reinforcement learning due to data contamination. arXiv preprint arXiv:2507.10532, 2025.

Wu, Z., Peng, B., et al. SymCode: Solving mathematical problems with code and self-debug. arXiv preprint arXiv:2410.02798, 2024.

Xiong, M., Hu, Z., Lu, X., Li, Y., Fu, J., He, J., and Hooi, B. Can LLMs express their uncertainty? An empirical evaluation of confidence elicitation in LLMs. In ICLR, 2024.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., et al. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. arXiv preprint arXiv:2409.12122, 2024.

Zhang, W. et al. PROVE: Verifying llm mathematical reasoning via code translation. arXiv preprint, 2024.

Zhang, Z., Zheng, C., Wu, Y., Zhang, B., Lin, R., Yu, B., Liu, D., Zhou, J., and Lin, J. The lessons of developing process reward models in mathematical reasoning. arXiv preprint arXiv:2501.07301, 2025.

Zheng, C., Zhang, Z., Zhang, B., Lin, R., Lu, K., Yu, B., Liu, D., Zhou, J., and Lin, J. ProcessBench: Identifying process errors in mathematical reasoning. arXiv preprint arXiv:2412.06559, 2024.

Zhou, J. P., Staats, C. E., Li, W., Szegedy, C., Weinberger, K. Q., and Feng, Y. Don't trust: Verify – grounding LLM quantitative reasoning with autoformalization. ICLR, 2024.

Appendix

The appendix supplements the main text with hyper-reproducible implementation details, full statistical methodology, the pre-registration text, additional results that did not fit in the eight-page main body, representative verifier case studies, and a broader-impact discussion. Section numbering follows the ICML appendix convention (A, B, C, …).

A. Extended Methods

A.1. Solver and extractor configuration

The base solver in every result reported in the main body is Qwen2.5-7B-Instruct-Turbo accessed via the Together AI inference endpoint. A dedicated control series additionally rotates the solver to DeepSeek-V3.1 (Section D.2). For the false-acceptance analysis of self-verification (Section 3.1 of the main text) the solver is Qwen2.5-Math-7B-Instruct, the contamination-flagged variant from Wu et al. (2025). Solver sampling parameters are held fixed across all benchmarks: temperature 0 for the plurality candidate; temperature 0.7, top-p 0.95, k=10 samples for entropy-based and self-consistency baselines; max_tokens=2048; system prompt empty. The chain-of-thought prompt template terminates with the literal string "The final answer is $\boxed{}$.".

The cross-family extractors are Llama-3.3-70B-Instruct-Turbo (Together AI), DeepSeek-V3 (Together AI), gpt-oss-120b (Together AI), claude-sonnet-4-6 (Anthropic), Qwen3-Coder-480B-A35B-Instruct (Together AI), and gpt-5-mini (OpenAI). Extractor sampling uses temperature 0.2, max_tokens=2048, and a system prompt fixing the output format to a single triple-backtick Python code block. The full extractor prompt is reproduced below; it is identical for every extractor and was frozen on 2026-04-01 prior to the n=498 scale-up.

Listing 2. Extractor prompt template (frozen 2026-04-01).

```
You are a mathematics verification engineer. You will be shown a
math problem. Your job is to write a Python function

    def verify(answer) -> bool:

that returns True if ‘answer‘ (a Python int, float, or sympy
expression) satisfies the problem’s constraints, and False
otherwise.

Hard rules:
  1. Do NOT solve the problem. Do NOT compute the answer in
     any form. Do NOT write ‘answer == <some literal>‘.
  2. The function must re-derive the problem’s constraints
     from the problem statement (algebraic equations,
     combinatorial conditions, geometric constraints, etc.)
     and check that ‘answer‘ satisfies them.
  3. You may import sympy. itertools, math, fractions.Fraction,
     and collections.Counter are pre-imported in the sandbox.
  4. Wall-clock time for a single call must be < 10 seconds.
  5. If the problem requires a proof, a construction, an
     existence argument, or otherwise has no bounded answer
     check, return EXACTLY the string ‘UNVERIFIABLE‘ (no quotes,
     no code block) and nothing else.
  6. The total script must be under 60 lines.

Now write the function for the following problem:

<<<PROBLEM>>>
```

A.2. Sandbox implementation

Each emitted script is executed in a Python subprocess sandbox with the following constraints. (i) SIGALRM-based wall-clock timeout of ten seconds per verify( ) invocation. (ii) Exception trapping around the entire call: any Exception or SystemExit maps to a False return rather than crashing the parent process. (iii) The pre-imported standard library is restricted to sympy, itertools, math, fractions.Fraction, and collections.Counter. (iv) Network sockets are not available in the subprocess. (v) The candidate $\hat{a}$ is cast to a SymPy sympify( ) expression before invocation; on SympifyError the candidate is passed as a Python string. The sandbox harness is approximately 80 lines of Python and runs on a single laptop CPU; no GPU is required.

A.3. Adversarial probe selection and rationale

The deployment-time adversarial probe set $\mathcal{P}(\hat{a}) = \{\hat{a}-1, \hat{a}+1, \hat{a}+7, \hat{a}\times 2\}$ was selected on a 50-problem held-out development split of MATH-500 (strata-balanced, disjoint from any test set used in the paper) and locked in the pre-registration document (Section C.3). The four perturbations target the four verifier failure modes observed during pilot studies: trivial return True verifiers (caught by any perturbation), additive-bias verifiers (caught by ± 1), modular-residue verifiers (caught by $+7$, chosen because 7 is coprime with $\{2, 3, 4, 5, 6\}$ and so triggers nontrivial residue changes), and multiplicative-scaling verifiers (caught by $\times 2$). The set was not expanded after locking; the appendix and supplementary code show the version frozen on 2026-04-01. For non-integer $\hat{a}$ the deterministic fallback set $\{0, 1, -1, 42, 100\} \setminus \{\hat{a}\}$ is used instead, again as fixed by pre-registration.

A gold-relative variant of the same probe set, $\{a-1, a+1, a+7, a\times 2\}$, is used only inside the benchmarking pipeline (where ground truth is available) to compute the "Adv-FP" column reported in Table 3 of the main text; this column measures the verifier's behaviour at the gold answer and is purely diagnostic. The deployment-time filter never accesses a.

A.4. Cross-extractor consensus protocol

For each problem x and each extractor E_i, the protocol records four values: (i) whether the extractor returned a syntactically valid Python script, (ii) whether the script produced a verify( ) function that imports successfully in the sandbox, (iii) whether the function survives the deployment-time adversarial probe, and (iv) whether the function accepts $\hat{a}$. A working verifier for problem x under extractor E_i is one for which (i)–(iii) all hold. Strict consensus accepts $\hat{a}$ only when every participating extractor produces a working verifier on x and every working verifier accepts. Loose consensus requires at least one working verifier across the participating extractors and unanimous acceptance among the working ones; problems where no extractor produces a working verifier are abstained on under either rule. The main text reports strict consensus everywhere unless explicitly stated otherwise.

A.5. Process Reward Model baseline implementation

Qwen2.5-Math-PRM-7B and Skywork-o1-Open-PRM-Qwen-2.5-7B were run on Modal A100 instances using the model cards' published step-segmentation regex. For each problem and each of the ten cached solver samples, the per-step rewards were aggregated using min-step, mean-step, product-step, and last-step aggregations as documented on the respective model cards; the problem-level score is the maximum across the ten samples. Each PRM is thresholded to match the X-SGRV top-tier coverage on each benchmark exactly (i.e. fixing the number of accepts k). The Skywork-PRM torch_dtype was set to torch.bfloat16; on FP32 the model silently exhausted the A100's 40 GB of GPU memory and produced NaN scores for sequences longer than ∼1,500 tokens, an issue not reported in the model card. The PRM scoring protocols were validated against published ProcessBench (Zheng et al., 2024) numbers on a 48-problem ProcessBench subset, reproducing the published values within 5 macro-F1 points on both PRMs.

A.6. Verifier classification protocol

A verifier is classified as working if it (a) accepts the gold answer a, (b) rejects each of four adversarial wrong candidates $\{a-1, a+1, a+7, 42\}$ (a fixed set, distinct from the deployment-time probe set in Section A.3), (c) halts within ten seconds on every input, and (d) does not raise an unhandled exception. A verifier that accepts a but also accepts some adversarial wrong candidate is classified as trivial-or-broken. A verifier that fails to accept a is classified as wrong-spec. The breakdown across MATH-175 for each extractor appears in Table 5.

B. Datasets and Benchmarks

B.1. MATH-500 stratified subsampling

The stratified n=175 subsample of MATH-500 contains exactly 35 problems from each of the five MATH difficulty levels, drawn by random seed 42 from the original 500 problems of Lightman et al. (2024). Index lists are deposited in data/math500_strat_175.json in the supplementary code release. The full n=500 run uses every problem;

two problems failed solver timeouts and the analysis sample size is therefore n=498.

B.2. AIME 2025 and CleanMath combo

AIME 2025 problems were obtained from the MathArena(Balunović et al., 2025) release on 2026-03-22; both AIME-I and AIME-II were used, totaling 30 problems. The CleanMath combo aggregates four post-cutoff competitions also from MathArena: HMMT-February 2025 (n=30), BRUMO 2025 (n=30), SMT 2025 (n=53), and APEX 2025 (n=12); total n=125. Every problem in this combo was released after both 2024-12-31 (the Qwen2.5 training cutoff) and 2025-01-01, and was checked against the contamination protocol of Wu et al. (2025): all 125 problems failed the verbatim-prefix test, confirming they are not in the solver's training distribution.

B.3. Stress benchmarks

Omni-MATH hard-100. The 100 hardest problems from Omni-MATH (Gao et al., 2024) stratified by the published difficulty score (range 9.0–9.5 of 10). PutnamBench-100. A 100-problem random sample of the public PutnamBench release (Tsoukalas et al., 2024), drawn by random seed 42; problems are mostly proof-mode and the prompt explicitly enables the UNVERIFIABLE refusal token.

C. Statistical Methodology

C.1. Confidence intervals

Every reported precision is accompanied by a 95% Clopper-Pearson exact binomial confidence interval, computed via scipy.stats.binomtest( ).proportion_ci(method="exact") in SciPy 1.13. Bootstrap confidence intervals on AUROC use 1,000 resamples drawn with replacement from the per-problem score distribution.

C.2. Hypothesis tests

Matched-binary comparisons between two methods on the same set of problems use McNemar's exact mid-p test(?) as implemented in statsmodels.stats.contingency_tables.mcnemar(exact=True). The 2×2 ablation comparing solution-grounded versus specification-grounded extraction (Section 3.2 of the main text) uses Fisher's exact test on the contingency table $(44, 0; 276, 334)$, yielding $p < 10^{-8}$.

C.3. Pre-registration document

The pre-registration text below was committed to the project repository on 2026-04-01 prior to the n=498 scale-up. The git commit hash is recorded in the anonymized supplementary code release.

> Pre-commit, n=498 MATH-500 cross-extractor consensus. Hypothesis. Strict two-way consensus (Llama-3.3-70B ∩ DeepSeek-V3) achieves top-tier precision ≥ 99% on the full MATH-500. Branches. A: precision ≥ 99%, claim publishable result. B: 95% ≤ precision < 99%, claim "high but imperfect" result with explicit precision number; do not overclaim. C: precision < 95%, audit each false acceptance by hand and report ablations; do not claim a top-tier selective-prediction signal. Halt-and-audit: precision < 90%, halt the line of work pending a structural fix. Locked parameters. Adversarial probe set $\{\hat{a} \pm 1, \hat{a} + 7, \hat{a} \times 2\}$ for integer $\hat{a}$, fallback set $\{0, 1, -1, 42, 100\} \setminus \{\hat{a}\}$ for non-integer $\hat{a}$. Sandbox timeout 10 seconds. Working-verifier definition fixed per Appendix A.6. Outcome posted before run. The result, if it falls in branch A, will be reported as a single number with a Clopper-Pearson confidence interval; if in branches B–C, additional ablations will be reported per branch specification.

The realized result was $171/171 = 1.000$, 95% Clopper-Pearson confidence interval $[0.979, 1.000]$, fulfilling Branch A.

D. Extended Results

D.1. Per-extractor verifier classification on MATH-175

Table 5 reports the per-extractor verifier-class breakdown across the stratified MATH-175 sample. The "trivial-or-broken" column equals the proportion of any emitted script that the deployment-time adversarial probe rejects,

and is therefore the headline rate at which the deployment-time filter (Section A.3) would intervene.

Table 5. Per-extractor verifier classification on MATH-175. "Working" satisfies the four-condition definition in Section A.6. "Wrong-spec" rejects the gold answer. "Trivial-or-broken" accepts the gold but also accepts at least one adversarial wrong candidate. "UNVERIFIABLE" is the explicit refusal token. Total: n=175 per row.

Extractor	Working	Wrong-spec	Trivial-or-broken	Unverif.
Llama-3.3-70B	119	21	13	22
DeepSeek-V3	124	18	11	22
Claude Sonnet 4.6	112	23	18	22
gpt-oss-120b	98	30	24	23
Qwen3-Coder-480B	105	27	19	24
GPT-5-mini	102	29	21	23

D.2. Solver rotation control

A solver-rotation control swapped the 7B Qwen solver for DeepSeek-V3.1 on the same cached extractor scripts, holding the verifier pipeline fixed. On the CleanMath combo, top-tier accepts rose from 4 of 4 to 15 of 15; precision remained at 1.000. This isolates the verifier's selectivity from the solver's accuracy: the verifier admits more candidates because the rotated solver answers correctly on more problems, not because the verifier behaves differently.

D.3. Lab-agnostic five-way consensus

The lab-agnostic test on stratified MATH-175 was constructed by selecting one frontier extractor from each of five organizationally distinct labs: Meta (Llama-3.3-70B), DeepSeek (DeepSeek-V3), OpenAI (gpt-oss-120b), Anthropic (Claude Sonnet 4.6), and Alibaba (Qwen3-Coder-480B). The accepted-tier size shrinks monotonically with each additional extractor, from 73 (single, Llama) $\rightarrow$ 72 (Llama $\cap$ DeepSeek) $\rightarrow$ 69 (3-way) $\rightarrow$ 69 (4-way) $\rightarrow$ 69 (5-way), and precision remains at 1.000 at every step. Adding GPT-5-mini as a sixth extractor reduces the accepted tier to 67 and preserves precision at 1.000.

D.4. Putnam scope test

On a 100-problem random sample of PutnamBench, the proof-mode prompt permitted explicit UNVERIFIABLE refusals. Extractor refusal rates were: Claude Sonnet 4.6 51/100, GPT-5-mini 35/100, DeepSeek-V3 44/100, Llama-3.3-70B 39/100. Among non-refused problems, the proportion that produced a working verifier accepting the gold answer was: Claude 19/49, GPT-5-mini 8/65, DeepSeek 14/56, Llama 11/61. The relatively high Claude refusal rate is consistent with explicit self-knowledge of proof-mode scope; this is consistent with prior observations that anthropic-trained models exhibit higher refusal rates on out-of-distribution mathematical tasks.

D.5. Lean compile-rate audit

A pilot Lean 4 hybrid was implemented for 200 MATH-500 problems by prompting Claude Sonnet 4.6 to emit, alongside the SymPy verifier, a Lean 4 theorem statement of the form theorem problem : <constraint> along with a placeholder by sorry proof. The compile rate against Mathlib4 4.10.0 was 158/200 = 78.9%; failures were predominantly import resolution errors and unicode-symbol parsing issues, not genuine type-mismatch errors. This rate is reported only as evidence that a hybrid SymPy-plus-Lean filter (proposed as future work in Section 4 of the main body) is now tractable on commodity API budgets; it is not used in any X-SGRV result.

D.6. Verifier triviality audit details

The static-pattern audit reported in main text Section 3.8 used the following classification rules. Trivial literal: regex ^\s*return\s+answer\s*==\s*[-0-9]+\s*$ anywhere in the body. Compound: at least two top-level and/or/if clauses or at least one for loop with a conditional. Bounded enumeration: at least one for loop iterating a fixed-size range or itertools product. Single constraint: a single arithmetic expression compared via == or $\leq/\geq$. The audit script is in src/audit/triviality_audit.py of the supplementary code release.

E. Verifier Case Studies

E.1. Working compound verifier (MATH-500, Level 4 algebra)

Problem prefix (lightly elided): "Find the product of all real roots of $x^2 - 5x + 6 = 0$."

```
def verify(answer):
    try:
        x = sympy.symbols('x')
        roots = sympy.solve(x**2 - 5*x + 6, x)
        product = sympy.prod(roots)
        return sympy.simplify(sympy.sympify(answer) - product) == 0
    except Exception:
        return False
```

E.2. Working enumeration verifier (CleanMath/HMMT-Feb 2025)

```
def verify(answer):
    try:
        n = int(str(answer).strip())
        count = 0
        for a in range(1, 10):
            for b in range(0, 10):
                for c in range(0, 10):
                    if (a + b + c) == 12 and a*b*c % 7 == 0:
                        count += 1
        return n == count
    except Exception:
        return False
```

E.3. Broken verifier rejected by the deployment-time filter

The following script accepts the gold answer but also accepts $\hat{a} + 7$ and is therefore rejected by the deployment-time adversarial filter:

```
def verify(answer):
    try:
        return int(str(answer).strip()) % 7 == 0
    except Exception:
        return False
```

E.4. UNVERIFIABLE refusal example (PutnamBench)

Problem prefix: "Prove that for every prime $p > 3$, the integer $1^{p-1} + 2^{p-1} + \cdots + (p-1)^{p-1}$ is congruent to -1 modulo p." The extractor returned the literal token UNVERIFIABLE, which the sandbox harness mapped to an explicit abstain verdict.

F. Limitations and Broader Impact

F.1. Operational scope

X-SGRV provides a structurally-grounded selective-prediction signal, not a continuous-score uncertainty estimator. Top-tier coverage is single-digit on contamination-clean competition mathematics by design, because the structural verifiability of a competition problem is a property of the problem itself, not of the solver's confidence. On the hardest 100 problems from Omni-MATH (difficulty 9.0–9.5), zero working verifiers were produced, defining the operational ceiling on non-proof problems. On Putnam-style proof problems the system self-detects scope by emitting the UNVERIFIABLE token on roughly half of the 100-problem sample.

F.2. Reliance on a strong cross-family extractor

The empirical results require a $\geq$ 70B cross-family extractor; at the 7B scale the working-verifier rate on MATH-500 is below 15%. The asymmetric extractor/solver pairing is consistent with the centralized-governance pattern advocated in Wang et al. (2024c) and Xiong et al. (2024) but remains a deployment cost.

F.3. Independence assumption

The cross-extractor consensus argument relies on an independence assumption between extractor families. The empirical 0/171 at the n=498 scale places a lower bound on effective independence, but is not a structural proof. A falsification protocol — measuring the empirical pairwise correlation between extractor verifier outputs and comparing to the independence-bound prediction — is proposed in Section 4 of the main text.

F.4. Broader impact

The deployment-time signal proposed here is intended to support abstention on uncertain mathematical reasoning, not to substitute for human review. In particular, deploying X-SGRV in high-stakes settings (e.g., automated grading, safety-critical scientific computation, or medical decision support) without an explicit human-in-the-loop fallback would invert the safety pattern the system is designed to support: the right deployment posture is to escalate, not commit, when the verifier abstains. Top-tier acceptances on contamination-clean benchmarks (e.g., 4 of 4 on the CleanMath combo) are correct in our experiments but the sample size is small and a deployment policy that treats individual accepts as authoritative without a human review tier would be misuse. The signal is also not a substitute for proof verification: UNVERIFIABLE is the correct verdict for the majority of proof-mode Putnam problems, and a Lean-based hybrid is the intended path for that regime.

F.5. Energy and compute footprint

The full n=498 MATH-500 scale-up consumed approximately 3.1×10^6 output tokens across all extractor and solver calls, equivalent to roughly \$45 in API costs at 2026-04 list prices. The PRM baselines required approximately 4 A100-hours on Modal. No model fine-tuning was performed in this work.