SELF-BRIER AND FORMAL SEMANTIC ENTROPY

Anonymous authors
Paper under double-blind review

ABSTRACT

An LLM may give conflicting answers to the same question while reporting high confidence in each. We introduce two uncertainty scores that summarize its expressed certainty across repeated outputs without reference answers. Self-Brier replaces the observed outcome in the Brier score with an indicator of agreement with the most frequent sampled answer. It averages the squared differences between confidence and this indicator, combining doubt among supporters with confidence among dissenters. Formal Semantic Entropy groups formalized claims by logical equivalence and computes entropy from the group frequencies. An auxiliary model translates responses into formulas over a shared vocabulary. A theorem prover, here the SMT solver Z3, checks equivalence under shared assumptions. The groups depend on which claims the translation retains and how faithfully it represents the response. We study three models on four public reasoning benchmarks. Formal entropy covers FOLIO, ZebraLogic, and GSM-Hard. In a secondary error-detection evaluation, Self-Brier has higher mean AUROC than voting on all four benchmarks, but its gains over confidence remain uncertain. Formal entropy has higher mean AUROC than our binary LLM-equivalence baseline on all three formalized benchmarks; confidence has higher mean AUROC than formal entropy on each.

1 INTRODUCTION

A confidence report describes a single response. To characterize an LLM's expressed certainty, we also need to know how its answers and explanations vary across repeated responses to the same question. A model may give conflicting answers while reporting high confidence in each, or repeatedly give the same answer with low confidence. Its explanations may also make different claims even when the final answers agree.

Uncertainty signals guide document retrieval in language generation (Jiang et al., 2023) and requests for human help in robot planning (Ren et al., 2023). Prior work has studied LLM uncertainty through verbalized confidence and agreement across repeated responses (Tian et al., 2023; Xiong et al., 2024). Reported confidence can improve answer selection when used to weight sampled answers (Taubenfeld et al., 2025). Semantic entropy addresses a further difficulty: differently worded responses can express the same meaning (Kuhn et al., 2023). It has been evaluated for detecting errors in factual answers and generated biographies (Farquhar et al., 2024). Extensions use it to guide retrieval (Zubkova et al., 2025) and detect hallucinations in medical visual question answering (Liao et al., 2025). Mean confidence does not show which answers receive high confidence. Answer frequencies also leave out differences in the claims made in the accompanying explanations.

We introduce two uncertainty scores for these aspects of repeated responses. *Self-Brier* replaces the observed outcome in the Brier score (Brier, 1950) with agreement with the most frequent, or modal, sampled answer. The agreement indicator is one if the answer matches the mode and zero otherwise. We average the squared difference between this indicator and each sample's reported confidence. The score combines doubt among supporters of the modal answer with confidence among dissenters. Once answers and confidences are available, it requires no additional model calls.

Formal Semantic Entropy uses logical equivalence to group formalized claims from sampled answers and explanations. An auxiliary model translates each response into a formula over a shared vocabulary, retaining its answer and selected claims from its explanation. A theorem prover checks

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.

whether the formulas are equivalent under shared assumptions. Our implementation uses the SMT solver Z3 (De Moura & Bjørner, 2008). We compute entropy from the resulting group frequencies. The groups depend on the accuracy of the translation and on which claims it retains.

We study Claude Sonnet 4.6, Qwen3-235B-A22B, and GPT-OSS-20B on OlympiadBench, FOLIO, ZebraLogic, and GSM-Hard. Each model receives the same 100 questions per benchmark, with ten sampled responses per question. Self-Brier covers all four benchmarks. Formal Semantic Entropy is evaluated on FOLIO, ZebraLogic, and GSM-Hard. Each score is compared with confidence, voting, answer entropy, and entropy based on LLM equivalence judgments, using the same sampled responses.

Neither score is a calibrated probability of correctness. As secondary validation, we test whether higher scores identify incorrect modal answers and whether retaining lower-scoring answers reduces error. Self-Brier has higher mean error-detection AUROC than voting on all four benchmarks, but its gains over confidence remain uncertain. Formal entropy has higher mean AUROC than our binary LLM-equivalence baseline on all three formalized benchmarks; confidence has higher mean AUROC than formal entropy on each. The formal results also depend on the translation protocol, failure handling, and the baseline prompt.

2 Problem Setup

We study uncertainty estimation from repeated responses to the same question. The goal is to summarize two observable features: how reported confidence aligns with answer agreement, and how the claims in explanations vary across responses. For a question x, a fixed model produces n samples (r_i, a_i, c_i), where r_i is a reasoning trace, a_i an answer, and $c_i \in [0, 1]$ the verbalized confidence in that answer. The scores use these responses and the question, without consulting a reference answer.

Let A be the indices with a parsed answer. We group these answers using normalized labels for categorical tasks and symbolic equivalence for mathematics. The most frequent class defines the modal prediction $\hat{a}$; ties go to the class first observed. Let $V \subseteq A$ contain the indices with both a parsed answer and confidence. Voting uses A; confidence-based scores use V. If the required set is empty, the corresponding score is unavailable.

All methods use the same sampled responses and fixed modal prediction. Reference answers enter only when we evaluate whether the scores identify incorrect modal predictions and support selective prediction (Section 4). Low uncertainty does not guarantee correctness.

3 Two Uncertainty Signals

3.1 Self-Brier

For each sample $i \in V$, define agreement with the modal answer as $y_i = \mathbf{1}[a_i \sim \hat{a}]$. The answer classes are the same as those used for voting. Self-Brier is

$$B_s(x) = \frac{1}{|V|} \sum_{i \in V} (c_i - y_i)^2 = \underbrace{\frac{1}{|V|} \sum_{i \in V : y_i = 1} (1 - c_i)^2}_{\text{supporter term}} + \underbrace{\frac{1}{|V|} \sum_{i \in V : y_i = 0} c_i^2}_{\text{dissenter term}}. \tag{1}$$

The first term grows when supporters of the mode express doubt. The second grows when dissenters express confidence. Both terms use the same denominator, so their sum is exactly the full score.

Figure 1 illustrates confident agreement, hesitant agreement, and confident dissent. When seven of ten samples agree with the mode and every sample reports confidence 0.95, Self-Brier is 0.2725. Mean-confidence uncertainty is 0.05.

If all samples give the same wrong answer with confidence 0.95, Self-Brier is 0.0025. The indicator y_i comes from the sampled answers, so the squared residual does not make Self-Brier a proper scoring rule for correctness. With constant confidence c and supporter fraction p,

$$B_s = (1 - c)^2 + (1 - p)(2c - 1). \tag{2}$$

When $c > 1/2$, dissent increases the score. It has no effect at $c = 1/2$ and decreases the score when $c < 1/2$. At a fixed confidence level, the vote fraction determines the score. When confidence

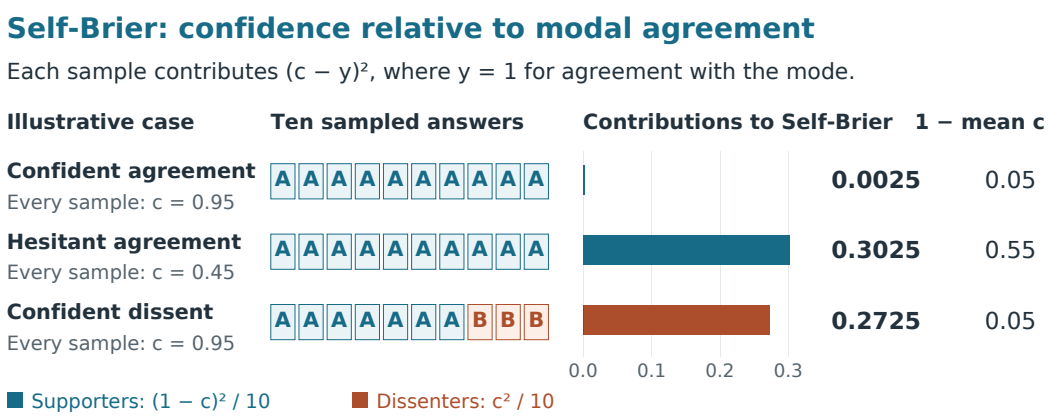


Figure 1: Self-Brier on three illustrative sets of ten samples. Letters denote sampled answer classes; colors indicate agreement with the mode. The stacked bars show supporter and dissenter contributions with a common denominator of ten. Confident dissent increases Self-Brier even when mean-confidence uncertainty remains low.

varies across samples, Self-Brier can distinguish sets with identical mean confidence and answer frequencies by whether high confidence accompanies agreement or dissent (Figure 7). Once answers have been grouped and confidences parsed, computing Self-Brier takes linear time in n and requires no further model calls.

3.2 Formal Semantic Entropy

Formal Semantic Entropy measures diversity among the formalized claims of sampled responses. The method has three steps: translate answers and explanations into formulas, group logically equivalent formulas, and compute entropy over the resulting groups. Responses can share a final answer yet fall into different groups because of an additional claim in an explanation.

Formal representation. An auxiliary model constructs a vocabulary shared by all responses to a question. It translates each response into a formula ϕ_i containing its answer and selected claims from its explanation, which we call its *retained commitments*. Let Γ denote any shared background assumptions from the question. The representation $\Gamma \wedge \phi_i$ records these commitments rather than the complete derivation.

Logical equivalence. Two formulas are equivalent under Γ if each entails the other under those assumptions. Equivalently,

$$\phi_i \equiv_\Gamma \phi_j \quad \Longleftrightarrow \quad \begin{cases} \Gamma \wedge \phi_i \wedge \neg\phi_j \text{ is unsatisfiable,} \\ \Gamma \wedge \phi_j \wedge \neg\phi_i \text{ is unsatisfiable.} \end{cases} \tag{3}$$

A satisfiable result in either direction provides a counterexample; two unsatisfiable results establish equivalence. Adding a claim already implied by Γ does not create a new equivalence class.

Clustering and entropy. We connect samples whose formulas are established as equivalent and take connected components as clusters. Samples with failed formalizations remain separate singletons; Section 4 specifies the validity checks. The clusters $\mathcal{C}$ partition all n samples. Each cluster C has relative frequency $\hat{p}(C) = |C|/n$. These frequencies are nonnegative and sum to one, defining an *empirical discrete distribution* over the observed clusters.

Formal Semantic Entropy is the Shannon entropy of this distribution, in bits:

$$H_{\text{formal}}(x) = -\sum_{C \in \mathcal{C}} \frac{|C|}{n} \log_2 \frac{|C|}{n}. \tag{4}$$

The entropy is zero for one cluster and $\log_2 n$ for n singleton clusters. Higher values indicate greater dispersion across the formal clusters. Figure 3 illustrates how entropy increases as a fixed number of

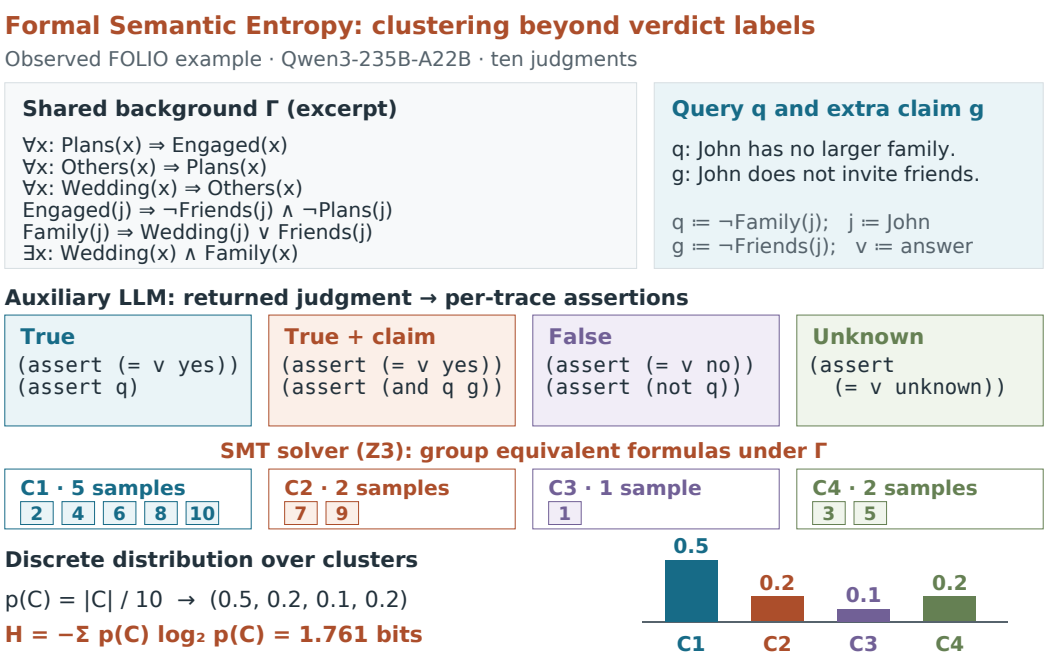


Figure 2: A recorded FOLIO question from Qwen3-235B-A22B. Both C_1 and C_2 return True; C_2 additionally retains the claim that John does not invite friends. This claim is consistent with Γ but not entailed by C_1 under Γ, so Z3 keeps the formulas separate. All ten formalizations are valid. Cluster probabilities $(0.5, 0.2, 0.1, 0.2)$ give 1.761 bits, whereas verdict-label entropy is 1.157 bits. Predicate names and formulas are abbreviated; displayed formulas are equivalent to the stored formulas under the full Γ.

clusters becomes more evenly populated. The entropy definition does not require a particular solver; Appendix E discusses other reasoning engines.

Worked example. Figure 2 illustrates the three steps for one evaluated FOLIO question. Two clusters give the same verdict but differ in one retained claim. Let ϕ_{C_i} represent cluster C_i. Applying Equation 3, the matrix gives the solver result for $\Gamma \wedge \phi_{C_i} \wedge \neg\phi_{C_j}$, with antecedents in rows and consequents in columns:

	C_1	C_2	C_3	C_4
C_1	unsat	sat	sat	sat
C_2	unsat	unsat	sat	sat
C_3	sat	sat	unsat	sat
C_4	sat	sat	sat	unsat

An `unsat` entry establishes implication; equivalence requires `unsat` in both directions. Here C_2 entails C_1, but not conversely. The extra claim in C_2 distinguishes the two True clusters.

What the solver establishes. Equation 3 establishes equivalence between the generated formulas under Γ. It does not certify the correctness of Γ, the sampled answer, or the translation. If a translation omits relevant claims, different arguments may receive equivalent formulas. If it adds distinctions absent from the text, equivalent arguments may receive different formulas. Our comparison with LLM entropy changes both the representation and the equivalence judge, so it evaluates the complete pipelines.

4 Empirical Study

Models and questions. We use Claude Sonnet 4.6, Qwen3-235B-A22B, and GPT-OSS-20B. The benchmarks cover mathematics (the English text-only subset of OlympiadBench (He et al., 2024)),

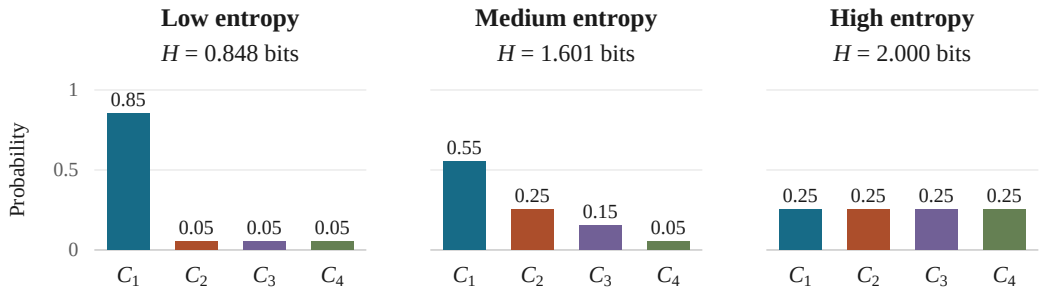


Figure 3: Constructed examples with 20 samples and four occupied clusters each. Entropy increases as cluster frequencies become more even; the uniform distribution reaches the four-cluster maximum of 2 bits.

natural-language logical inference (FOLIO (Han et al., 2024)), constraint puzzles (ZebraLogic (Lin et al., 2025)), and arithmetic word problems (GSM-Hard (Gao et al., 2023)). Each model has ten sampled responses for each of 100 shared questions per benchmark. This gives 400 distinct questions and 1,200 model–question records. We implement formalization for FOLIO, ZebraLogic, and GSM-Hard, giving 900 formal-entropy records before filtering for score availability.

The models and benchmarks were selected retrospectively from an existing study based on model family and artifact availability. Every uncertainty method uses the same original responses within each model–question comparison. ZebraLogic has two prompt variants whose use differs across models. Appendix A describes the selection, prompts, and sampling settings; Appendix F gives the full prompt text.

Formalization and solver protocol. Translations must parse, be satisfiable with their background, and pass task-specific checks against the sampled answer. Failures become singleton clusters; an unavailable shared signature makes the score unavailable. Z3 checks Equation 3 in SMT-LIB, with five seconds per implication. Unresolved comparisons add no equivalence edge, with no LLM fallback. Appendix D specifies the encodings, validity checks, and repair limits; Appendix F supplies the templates and worked calculations.

Baselines. We use four baselines. Mean-confidence uncertainty is $1 - |V|^{-1} \sum_{i \in V} c_i$. Vote uncertainty is one minus the modal answer fraction among parsed answers. Label entropy uses the frequencies of the same answer classes.

LLM entropy groups responses using pairwise equivalence judgments from Claude Sonnet 4.5, the auxiliary model that also supplies the formalizations. This differs from the evaluated Sonnet 4.6 model. The prompt requires agreement in the conclusion and substantive argument. Responses judged equivalent are connected, and the connected components define the clusters. We compute entropy from the recorded cluster sizes. The main comparison uses a binary prompt with a 256-token response limit. A later prompt with a 4,096-token limit is assessed separately in Appendix C.

Evaluation measures. We use reference answers for a secondary evaluation of error detection and selective prediction. Error-detection AUROC measures how often an incorrect modal answer receives higher uncertainty than a correct one. Higher AUROC is better; tied uncertainty values receive half credit. For selective prediction, we retain answers in order of increasing uncertainty. Coverage is the fraction of scored questions retained, and risk is their error rate. We report the area under the risk–coverage curve (AURC); lower is better. All methods in a comparison use the same scored questions.

We calculate metrics separately for each model and benchmark. The main tables report the mean of the defined model-level AUROCs for each benchmark. AUROC is undefined when there are no errors or no correct answers; these cases remain visible but do not enter the mean. We use 2,000 paired bootstrap resamples of questions to estimate uncertainty in differences between methods. The same resampled question identities are used across models. The intervals are conditional on

the selected models and benchmarks. Appendix B explains how ties, eligibility, and resampling are handled.

Analyses of the scores. We examine Self-Brier's supporter and dissenter terms separately. The validity policy affects formal entropy because a response whose encoding contradicts the background can become a singleton before equivalence is tested. We therefore compare the score with the overall failure rate and the rate of answer-inconsistency failures alone, and examine questions for which all ten formulas pass the validity checks. The main results use raw entropy. A separate ablation applies the same confidence adjustment to formal, LLM, and label entropy.

To assess the effect of fewer samples, we draw 20 subsets of five responses from each question's original ten. For each draw, we recompute the modal answer, its correctness, and the uncertainty scores. Reported means and ranges summarize metrics across draws. The original LLM baseline lacks the pairwise records needed for this comparison and is evaluated only at ten samples. Appendix C gives the ablation and sampling protocols.

5 RESULTS

Self-Brier improves mean error-detection AUROC over voting, while formal entropy improves over the binary LLM-equivalence baseline. Confidence alone remains competitive (Tables 1 and 2). Entries average the model-level AUROCs within each benchmark.

The comparisons use 1,180 of 1,200 model–question records for Self-Brier and 894 of 900 for formal entropy. The 20 omitted Self-Brier records lack a usable answer–confidence pair: 14 GPT-OSS records on OlympiadBench, five Qwen records on ZebraLogic, and one GPT-OSS record on ZebraLogic. All excluded records are errors. The formal comparison omits the same six ZebraLogic records. Risk–coverage curves use only scored questions; missing scores are not counted as abstentions. Qwen and GPT-OSS each have only two scored errors on ZebraLogic.

Benchmark	n/errors	K	**Self-Brier**	Confidence	Vote	Label entropy	LLM entropy
OlympiadBench	286/113	3	0.710	0.650	0.698	0.693	0.532
FOLIO	300/48	3	0.761	0.768	0.708	0.702	0.701
ZebraLogic	294/11	3	0.908	0.910	0.821	0.820	0.492
GSM-Hard	300/61	3	0.817	0.817	0.734	0.733	0.602

Table 1: Error-detection AUROC for Self-Brier and four baselines. Entries average the defined model-level AUROCs within each benchmark. n/errors counts scored model–question records and their errors; K is the number of models with defined AUROC. Confidence denotes $1-\bar{c}$. All columns use the same questions.

Self-Brier. Self-Brier has higher mean AUROC than vote uncertainty on all four benchmarks (Table 1). On OlympiadBench, it averages 0.710, compared with 0.698 for voting and 0.650 for confidence. The paired difference from confidence is 0.060, with a 95% bootstrap interval of $[-0.011, 0.141]$. That difference is 0.164 for Sonnet, 0.028 for Qwen, and -0.013 for GPT-OSS. Confidence alone gives similar or higher mean AUROC on the other three benchmarks. All four benchmark-level intervals for the comparison with confidence include zero (Figure 4).

Benchmark	n/errors	K	**Formal entropy**	Confidence	Vote	Label entropy	LLM entropy
FOLIO	300/48	3	0.764	0.768	0.708	0.702	0.701
ZebraLogic	294/11	3	0.834	0.910	0.821	0.820	0.492
GSM-Hard	300/61	3	0.780	0.817	0.734	0.733	0.602

Table 2: Raw Formal Semantic Entropy and four baselines on the three formalized benchmarks. Counts and averaging follow Table 1. The LLM baseline judges the original text; formal entropy uses translated formulas and solver checks. Variants that use confidence to break ties appear separately in the appendix.

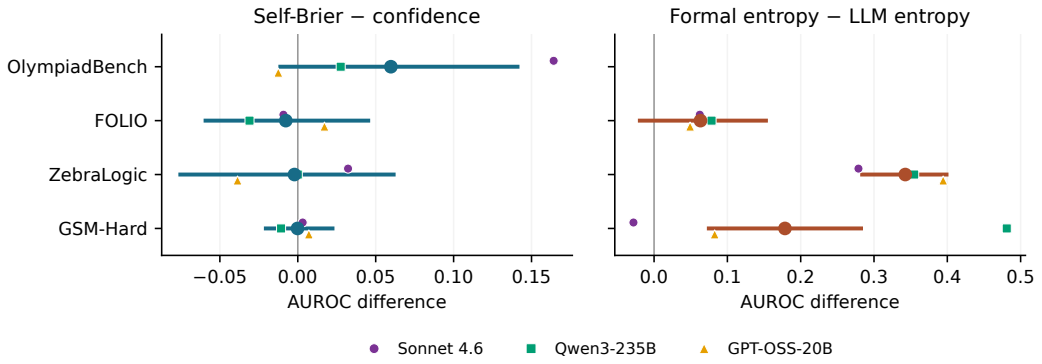


Figure 4: Paired AUROC differences on the same question sets as the main tables. Large points and horizontal bars show the model average and 95% question-bootstrap interval. Smaller purple circles, green squares, and gold triangles show Sonnet, Qwen, and GPT-OSS, respectively.

Formal Semantic Entropy. Formal entropy has higher mean AUROC than the original LLM baseline on FOLIO, ZebraLogic, and GSM-Hard (Table 2). The largest paired difference is 0.343 on ZebraLogic, with a 95% bootstrap interval of $[0.284, 0.399]$. On GSM-Hard, the difference is 0.179 with interval $[0.075, 0.282]$. FOLIO's interval includes zero. The GSM-Hard difference varies across models: $+0.481$ for Qwen, -0.028 for Sonnet, and $+0.083$ for GPT-OSS.

Confidence has higher mean AUROC than formal entropy on all three benchmarks: 0.768 versus 0.764 on FOLIO, 0.910 versus 0.834 on ZebraLogic, and 0.817 versus 0.780 on GSM-Hard. Formal entropy is close to label entropy on ZebraLogic and above label entropy's 0.733 on GSM-Hard. Appendix B gives the individual model results and risk–coverage curves.

6 Analysis

6.1 Confidence and answer agreement

Combining doubt among supporters with confidence among dissenters improves error ranking over either term alone on OlympiadBench and FOLIO (Figure 5). On OlympiadBench, the full score averages 0.710 AUROC, compared with 0.596 for the supporter term and 0.693 for the dissenter term. On FOLIO, the full score averages 0.761, compared with 0.743 and 0.703 for the two terms. Mean confidence still reaches 0.768 on FOLIO. Combining the terms can therefore improve on each component without improving on confidence alone.

The differences from confidence also vary by model. A post hoc check omitting GPT-OSS-20B retains both positive and negative benchmark-level differences (Appendix B).

6.2 Formalization and equivalence

Formalization is limited by the text available to translate. In GPT-OSS's GSM-Hard run, 996 of 1,000 saved outputs contain no text after removing answer and confidence tags. These outputs support answer encodings but supply no derivation. A model-assisted inspection of 60 trace–formula pairs identified examples of omitted claims, answer changes in failed repairs, and translations that retained little beyond the answer. Some flags concerned permissible omissions, so the inspection does not provide a measured fidelity rate (Appendix D).

On matched GSM-Hard responses, the current encoding raises mean formal AUROC from 0.713 to 0.780, a paired gain of 0.068 with interval $[0.027, 0.112]$. Sonnet and Qwen retain equations linking quantities in 995 and 990 formulas, respectively. Their gains over the earlier encoding are 0.021 and 0.048, with intervals including zero. GPT-OSS gains 0.134 after applying a rule that directly encodes the numerical answer when the stored response has no explanation. All 992 accepted GPT-OSS

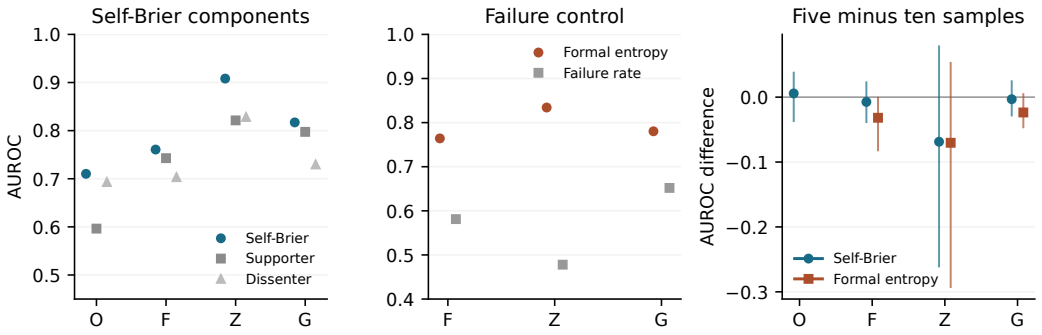


Figure 5: Effects of score components, formalization failures, and sample count. Left: Self-Brier and its components. Middle: formal entropy and failure rate. Right: change in AUROC when reducing the number of samples from ten to five. In the right panel, points and bars show the mean and range across 20 draws, with predictions recomputed at each sample count. O, F, Z, G denote OlympiadBench, FOLIO, ZebraLogic, and GSM-Hard. Sampling comparisons can include fewer questions and models than the main tables.

formulas encode only the parsed answer. The aggregate gain therefore cannot be attributed solely to richer algebra. Both protocols use identical responses, so Self-Brier is unchanged (Appendix F.3).

Failures affect entropy because each failed formalization becomes a singleton. Of 9,000 formalizations, 8,433 pass the validity checks. The other 567 comprise 351 unsatisfiable formulas, 196 format failures, six categorical answer conflicts, and 14 unsupported numerical answers. The unsatisfiable group includes 91 algebraic contradictions preserved without repair. All pairwise checks between accepted formulas are resolved.

Failure rate alone has lower mean error-detection AUROC than formal entropy on all three benchmarks. Restricting FOLIO to questions with ten valid formulas reduces the formal-entropy advantage over LLM entropy from 0.063 to 0.006. This restriction also changes the questions being evaluated, so it cannot isolate the contribution of failures.

A later LLM-equivalence prompt that requests confidence and permits a longer response narrows the gap from formal entropy on ZebraLogic (Appendix C). Adding mean confidence to break entropy ties gives label entropy an AUROC of 0.904 on ZebraLogic, compared with 0.835 for formal entropy with the same adjustment. These adjusted scores are separate from raw entropy. The measured advantage depends on the baseline prompt and on whether confidence is used to break ties.

6.3 Sampling and computational cost

Reducing the sample count can change both the modal answer and its uncertainty. Across 20 draws of five samples, mean formal AUROC falls by 0.032 on FOLIO, 0.070 on ZebraLogic, and 0.024 on GSM-Hard relative to ten samples. Self-Brier changes little on OlympiadBench, FOLIO, and GSM-Hard, but drops more on ZebraLogic and varies widely across draws. The comparisons use paired question sets; eligibility can differ from the main tables (Figure 5).

Self-Brier requires no further model calls once samples include confidence reports. Formal entropy requires a shared-vocabulary call, translations, and any repairs, although the GSM-Hard guard directly encodes outputs that contain only an answer. Recorded attempts average 7.71–13.53 auxiliary calls per question with ten samples across benchmarks, before cache reuse. The binary LLM baseline uses up to $\binom{10}{2} = 45$ calls and skips identical-text pairs. Formalization prompts are longer, so call counts alone do not establish token or latency savings. Median solver time is 0.119–0.200 seconds per question, excluding model inference, process startup, and scheduling (Appendix E).

7 Related work

Self-consistency selects an answer by voting across repeated reasoning samples (Wang et al., 2022). Later work examines its calibration (Wang et al., 2024) and weights votes by reported confidence (Taubenfeld et al., 2025). Self-Brier keeps the modal answer fixed and uses confidence and agreement to quantify uncertainty in the sampled responses. Its target is agreement within the sample, rather than the preference distribution used to train probabilistic autoraters (Li et al., 2025).

Semantic entropy groups outputs by meaning (Kuhn et al., 2023; Farquhar et al., 2024). We use its discrete cluster-frequency estimator (Farquhar et al., 2024), with equivalence between translated claims established by a solver. Sharma & David (2025) cluster generated programs using bounded symbolic execution to estimate uncertainty. Our LLM baseline judges agreement in the original conclusion and argument.

Logic-LM and SatLM translate language into symbolic problems that a solver can answer (Pan et al., 2023; Ye et al., 2023). In our method, the language model supplies the answers and the solver compares their formalized claims. Ganguly et al. (2026) study uncertainty in LLM-generated SMT-LIB artifacts using probabilistic grammars to describe their distribution. Formal Semantic Entropy estimates a distribution over clusters formed by logical equivalence, so formulas with different syntax can contribute to the same cluster.

Selective prediction studies how to trade coverage for lower error by withholding uncertain predictions (Geifman & El-Yaniv, 2017). We use risk–coverage curves to evaluate the scores after the modal answers have been selected.

8 Limitations

Neither score is a calibrated probability of correctness or a direct measure of internal beliefs. Confident unanimity can yield low Self-Brier even when every answer is wrong. Formal entropy measures distinctions among generated formulas: omitted or added claims, singleton failures, and unresolved solver checks can change it independently of variation in the original explanations. Solver checks do not establish translation fidelity, and our model-assisted inspection supplies no independently validated fidelity estimate.

The evaluation covers three models and 100 shared questions from each of four benchmarks, selected retrospectively by model family and artifact availability. Differences in prompts and sampling settings limit cross-model comparisons. Metrics are conditional on score availability; all excluded records are errors, and two models have only two scored errors each on ZebraLogic. These constraints limit the precision and generality of the observed differences.

The formal-versus-LLM comparison changes both representation and equivalence judgment, so it does not isolate the benefit of solver checks. Its outcome also depends on the baseline prompt and whether confidence breaks entropy ties. Formalization requires task-specific encodings, with no evaluated implementation for OlympiadBench. We have not measured the effort needed to extend those encodings to new tasks. Auxiliary translation and repair calls add cost, and the available records do not support a complete dollar or end-to-end latency comparison.

9 Conclusion

Self-Brier summarizes how reported confidence aligns with sampled answer agreement. Formal Semantic Entropy measures variation among the claims retained by formalization, using a solver to make the equivalence criterion explicit. Both scores aggregate observable aspects of expressed certainty without reference answers.

In the error-detection evaluation, Self-Brier has higher mean AUROC than voting on all four benchmarks, but gains over confidence remain uncertain. Formal entropy has higher mean AUROC than the binary LLM-equivalence baseline on all three formalized benchmarks, while confidence is higher on each. The formal results depend on translation, failure handling, and the baseline prompt. Better error ranking alone does not establish that the formulas faithfully represent variation in the original explanations.

REPRODUCIBILITY STATEMENT

Section 3 defines both scores; Section 4 specifies the formalization checks. Appendix A describes the data selection, saved samples, parsing rules, and generation settings, including gaps in the historical records. Appendices B and C specify the evaluation and resampling procedures. Appendix F provides the full prompts and worked calculations.

AI-USE DISCLOSURE

We used OpenAI Codex to assist with drafting and editing the manuscript, implementing methods conceived by the human authors, writing evaluation code, inspecting reasoning traces, and understanding failure modes. We did not use it to find related work, design the evaluation protocol, or sketch method figures. The human authors reviewed and checked all code, analyses, figures, and manuscript text and take responsibility for the final work.

REFERENCES

Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. cvc5: A versatile and industrial-strength SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 415–442. Springer, 2022. doi: 10.1007/978-3-030-99524-9_24.

Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950. doi: 10.1175/1520-0493(1950)078⟨0001:VOFEIT⟩2.0.CO;2.

Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340. Springer, 2008.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, pp. 625–635. Springer, 2021. doi: 10.1007/978-3-030-79876-5_37.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

Debargha Ganguly, Vikash Singh, Sreehari Sankar, Biyao Zhang, Xuecen Zhang, Srinivasan Iyengar, Xiaotian Han, Amit Sharma, Shivkumar Kalyanaraman, and Vipin Chaudhary. Grammars of formal uncertainty: When to trust llms in automated reasoning tasks. In *Advances in Neural Information Processing Systems*, volume 38, pp. 34213–34255, 2026.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799. PMLR, 2023. URL https://proceedings.mlr.press/v202/gao23f.html.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, et al. FOLIO: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22017–22031, 2024.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench:

A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.211.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7969–7992, 2023.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *arXiv preprint arXiv:2302.09664*, 2023.

Zhuohang Li, Xiaowei Li, Chengyu Huang, Guowang Li, Katayoon Goshvadi, Bo Dai, Dale Schuurmans, Paul Zhou, Hamid Palangi, Yiwen Song, Palash Goyal, Murat Kantarcioglu, Bradley A. Malin, and Yuan Xue. Judging with confidence: Calibrating autoraters to preference distributions. *arXiv preprint arXiv:2510.00263*, 2025. URL https://arxiv.org/abs/2510.00263.

Zehui Liao, Shishuai Hu, Ke Zou, Huazhu Fu, Liangli Zhen, and Yong Xia. Vision-amplified semantic entropy for hallucination detection in medical visual question answering. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 669–679. Springer, 2025.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. ZebraLogic: On the scaling limits of LLMs for logical reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 37889–37905. PMLR, 2025. URL https://proceedings.mlr.press/v267/lin25i.html.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, 2023.

Allen Z Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, et al. Robots that ask for help: Uncertainty alignment for large language model planners. In *arXiv preprint arXiv:2307.01928*, 2023.

Arindam Sharma and Cristina David. Assessing correctness in llm-based code generation via uncertainty estimation. *arXiv preprint arXiv:2502.11620*, 2025.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence improves self-consistency in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20090–20111. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.1030.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5433–5442, 2023.

Ante Wang, Linfeng Song, Ye Tian, Baolin Peng, Lifeng Jin, Haitao Mi, Jinsong Su, and Dong Yu. Self-consistency boosts calibration for math reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 6023–6029. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.349.

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. In *arXiv preprint arXiv:2203.11171*, 2022.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. In *International Conference on Learning Representations*, volume 2024, pp. 23650–23678, 2024.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. SatLM: Satisfiability-aided language models using declarative prompting. In *Advances in Neural Information Processing Systems*, volume 36, pp. 45548–45580, 2023.

Hanna Zubkova, Ji-Hoon Park, and Seong-Whan Lee. SUGAR: Leveraging contextual confidence for smarter retrieval. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

A INPUTS AND REPRODUCIBILITY

The evaluation is a retrospective subset of an existing study. We selected three model families and tasks with available artifacts, retaining both favorable and unfavorable cases. The subset is not an unseen-benchmark test. Original generations are retained; we do not replace difficult questions or models after reanalysis.

Inputs and artifacts. The evaluation uses the original ten sampled responses for each selected model–question record. The manifest lists input files, question hashes, sample hashes, prompt and code hashes, and the formulas used in the solver runs. It contains 400 distinct recovered question texts. Analysis code is organized into input preparation, formalization repair, solver checks, metric computation, artifact inspection, and figure generation. The protocol tests cover errors that can change the scientific comparison, including parsing a revised final answer, using mathematical equivalence consistently, missing confidence, recomputing five-sample correctness, and handling ties.

Formalization versions. All 900 active formal partitions use the same pair-comparison and singleton policy. The input audit checked the question, sampled text, and supplied answer against each stored formalization prompt. We regenerated 42 FOLIO and ZebraLogic records with mismatched questions or changed final-answer commitments; the remaining records on those tasks use their original formulas. All 300 GSM-Hard records use the algebraic translation protocol, which preserves contradictory commitments without semantic repair (Appendix F.3). The earlier numerical encodings remain a matched ablation. Historical entropy scores were not copied into the proposed methods' results.

LLM-baseline provenance. The binary LLM baseline's raw entropy is recomputed from the cluster sizes recorded in the original sampling run. Individual binary-oracle responses were not retained, so their verdicts cannot all be audited retrospectively. Appendix F.16 supplies the archived equivalence templates and describes the limits of their provenance.

Parsing. The confidence prompts specify percentages. The parser accepts bracketed numeric values and divides by 100, including interpreting literal `1` as 1%. Out-of-range or absent values are missing. We use the last complete answer tag, so an early tentative answer does not override a later revision. Categorical answers are normalized to the task's labels; mathematical expressions retain case and are grouped using symmetric checks from `math-verify` version 0.8.0. The same groups define the vote, its supporters, and label entropy. Ties use first appearance. Ground-truth matching uses the corresponding answer relation.

Sampling. We use ten sampled responses per question. The recorded sampling temperature is 0.7. Available settings specify a 16,384-token output limit, with 8,192 tokens for the original Qwen ZebraLogic run. Output limits were not recorded for every original request, so historical generation settings are only partially documented.

The recovered ZebraLogic prompts contain two wording variants. Sonnet uses a reasoning-first prompt on all 100 questions; GPT-OSS uses an initial-answer-commitment prompt on all 100. Qwen uses the former on 48 questions and the latter on 52. The parser uses the last complete answer tag for both variants. All uncertainty methods within a model–question comparison use the same outputs, but cross-model differences do not isolate model identity from prompt wording. Appendix F reproduces all four response and confidence templates, including the common mathematics and FOLIO templates.

We use Claude Sonnet 4.5 at temperature zero for vocabulary construction, formalization, and equivalence judgments. Vocabulary construction, translation, and repair use a 4,096-token output limit. The original binary equivalence baseline uses 256 tokens. A later similarity prompt with a 4,096-token limit is evaluated separately as a sensitivity analysis on six model–benchmark combinations.

Availability. Table 3 separates all questions from the scored subset. "Valid c_i" counts samples with both a usable answer and confidence. "No explanation" counts samples whose stored text is empty after removing answer and confidence tags. This is a syntactic description of the available

output, not a measurement of the model's internal reasoning. Missing outputs are not assigned zero uncertainty.

Benchmark	Model	All n	Errors	Scored n	Excluded errors	Valid c_i	No explanation
OlympiadBench	Sonnet 4.6	100	24	100	0	982	0
OlympiadBench	Qwen3-235B	100	43	100	0	955	0
OlympiadBench	GPT-OSS-20B	100	60	86	14	350	951
FOLIO	Sonnet 4.6	100	17	100	0	1000	0
FOLIO	Qwen3-235B	100	15	100	0	1000	0
FOLIO	GPT-OSS-20B	100	16	100	0	999	291
ZebraLogic	Sonnet 4.6	100	7	100	0	957	0
ZebraLogic	Qwen3-235B	100	7	95	5	899	1
ZebraLogic	GPT-OSS-20B	100	3	99	1	841	992
GSM-Hard	Sonnet 4.6	100	20	100	0	1000	0
GSM-Hard	Qwen3-235B	100	23	100	0	994	0
GSM-Hard	GPT-OSS-20B	100	18	100	0	992	996

Table 3: Question and sample availability. Each model–benchmark combination has ten samples per question. All 20 excluded records have incorrect modal answers under the recorded-output policy.

B Model-level results and selective prediction

The main tables use equal weight for the models with defined AUROC within a benchmark. They do not pool questions across benchmarks. Table 4 reports every selected combination. AURC in Table 5 and Figure 6 describes the same scored question sets. Differences in achievable risk make raw AURCs across models with different accuracies unsuitable as a common leaderboard.

Model sensitivity. In a post hoc check omitting GPT-OSS-20B, the Self-Brier AUROC difference from confidence averages $+0.096$ on OlympiadBench, -0.020 on FOLIO, $+0.016$ on ZebraLogic, and -0.004 on GSM-Hard. This preserves both positive and negative differences. These averages have no new interval estimates and do not identify which model properties account for the differences.

Risk and ties. For N scored questions ordered by uncertainty, we average risk over the N coverages k/N. In a tie block of size b containing e errors, the expected number of errors in its first j retained positions is je/b. This eliminates dependence on input row order. These curves are conditional on score availability; a deployment policy that abstains on missing outputs would have a different coverage denominator.

Benchmark	Model	n/errors	Self-Brier	Conf.	Vote	Label	LLM	Formal
OlympiadBench	Sonnet 4.6	100/24	0.798	0.634	0.862	0.855	0.521	—
OlympiadBench	Qwen3-235B	100/43	0.758	0.731	0.697	0.685	0.685	—
OlympiadBench	GPT-OSS-20B	86/46	0.574	0.587	0.535	0.538	0.389	—
FOLIO	Sonnet 4.6	100/17	0.777	0.787	0.657	0.654	0.647	0.709
FOLIO	Qwen3-235B	100/15	0.749	0.780	0.715	0.707	0.703	0.781
FOLIO	GPT-OSS-20B	100/16	0.755	0.738	0.753	0.746	0.753	0.802
ZebraLogic	Sonnet 4.6	100/7	0.882	0.849	0.798	0.801	0.608	0.886
ZebraLogic	Qwen3-235B	95/2	1.000	1.000	0.981	0.981	0.527	0.882
ZebraLogic	GPT-OSS-20B	99/2	0.843	0.881	0.683	0.678	0.340	0.735
GSM-Hard	Sonnet 4.6	100/20	0.882	0.879	0.753	0.754	0.830	0.802
GSM-Hard	Qwen3-235B	100/23	0.819	0.829	0.714	0.710	0.325	0.806
GSM-Hard	GPT-OSS-20B	100/18	0.750	0.743	0.734	0.734	0.651	0.733

Table 4: Per-model error-detection AUROC. A dash denotes an undefined metric or a method outside the evaluated scope.

Paired intervals. We resample question identities with replacement and use the same multiplicities for each model. Each comparison uses the same complete-case set as its main table. Models with undefined original AUROC remain excluded from its interval. If a resample lacks an error or success for an otherwise eligible model, we redraw. This conditions the interval on both classes remaining present; low-error results remain fragile. The intervals describe repeated questions from this fixed evaluation, not transfer to other benchmarks. No interval is interpreted as a simultaneous guarantee across all comparisons.

Benchmark	Model	n/errors	Self-Brier	Conf.	Vote	Label	LLM	Formal
OlympiadBench	Sonnet 4.6	100/24	0.124	0.196	0.084	0.086	0.237	—
OlympiadBench	Qwen3-235B	100/43	0.261	0.294	0.301	0.305	0.316	—
OlympiadBench	GPT-OSS-20B	86/46	0.460	0.468	0.529	0.528	0.612	—
FOLIO	Sonnet 4.6	100/17	0.066	0.064	0.124	0.124	0.126	0.104
FOLIO	Qwen3-235B	100/15	0.075	0.070	0.091	0.092	0.093	0.071
FOLIO	GPT-OSS-20B	100/16	0.073	0.074	0.078	0.079	0.079	0.063
ZebraLogic	Sonnet 4.6	100/7	0.012	0.015	0.028	0.028	0.042	0.011
ZebraLogic	Qwen3-235B	95/2	0.000	0.000	0.001	0.001	0.018	0.003
ZebraLogic	GPT-OSS-20B	99/2	0.004	0.003	0.013	0.014	0.023	0.007
GSM-Hard	Sonnet 4.6	100/20	0.055	0.056	0.114	0.114	0.063	0.095
GSM-Hard	Qwen3-235B	100/23	0.097	0.095	0.147	0.148	0.364	0.105
GSM-Hard	GPT-OSS-20B	100/18	0.099	0.099	0.103	0.103	0.125	0.103

Table 5: Per-model AURC; lower is better. Zero-error cells have zero risk at every scored coverage.

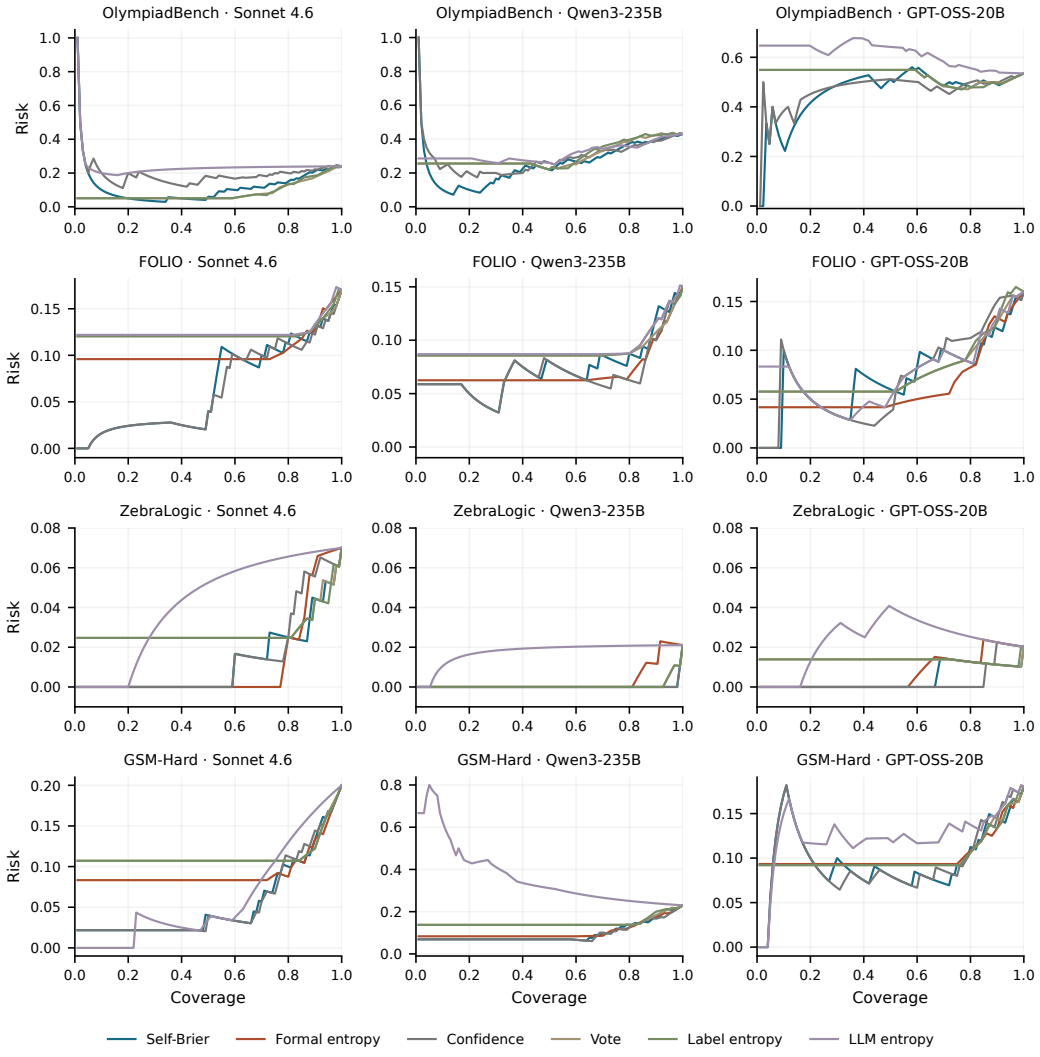


Figure 6: Risk–coverage curves on the common scored question sets. Coverage is relative to scored questions, not all generated questions.

Benchmark	ΔB_s	95% interval	ΔH_{formal}	95% interval
OlympiadBench	0.060	[-0.011, 0.141]	—	—
FOLIO	-0.008	[-0.059, 0.045]	0.063	[-0.019, 0.153]
ZebraLogic	-0.002	[-0.075, 0.062]	0.343	[0.284, 0.399]
GSM-Hard	0.000	[-0.020, 0.022]	0.179	[0.075, 0.282]

Table 6: Paired differences from confidence for Self-Brier and from binary LLM entropy for formal entropy, with percentile question-bootstrap intervals.

C Ablations

The component ablation keeps the denominator in Equation 1 fixed for both terms. Failure-rate uncertainty is the fraction of traces whose final status is not `ok`; answer-inconsistency uncertainty counts only that failure status. The all-valid analysis requires ten accepted formalizations and still uses the common score-availability set. It can preferentially remove difficult questions and is not an intervention on formalization quality.

Benchmark	n/errors	Self-Brier	Supporter	Dissenter	Confidence	Vote
OlympiadBench	286/113	0.710	0.596	0.693	0.650	0.698
FOLIO	300/48	0.761	0.743	0.703	0.768	0.708
ZebraLogic	294/11	0.908	0.821	0.828	0.910	0.821
GSM-Hard	300/61	0.817	0.797	0.730	0.817	0.734

Table 7: Self-Brier, its components, confidence, and voting. Entries are mean model-level error-detection AUROCs within each benchmark.

Benchmark	n/errors	Formal entropy	Failure rate	Answer inconsistency
FOLIO	300/48	0.764	0.581	0.492
ZebraLogic	294/11	0.834	0.478	0.500
GSM-Hard	300/61	0.780	0.652	0.500

Table 8: Formal entropy and failure-based controls, measured by mean model-level error-detection AUROC within each benchmark.

Confidence tiebreak. For each entropy H, the separate variant is $(1-\alpha)H + \alpha(1-\bar{c})$ with $\alpha = 10^{-4}$. The same coefficient and confidence definition apply to label, LLM, and formal entropy. The raw score remains the method definition. The coefficient primarily resolves discrete entropy ties; the formula is the exact implementation rather than an informal secondary sort.

The later prompt asks the equivalence judge to report confidence and raises its response limit from 256 to 4,096 tokens. Mean LLM-entropy AUROC changes from 0.492 to 0.706 on ZebraLogic and from 0.701 to 0.685 on FOLIO. The judge's confidence is not used for clustering and differs from the sampled model's confidence used in the tiebreak ablation. Appendix F.16 gives both prompts.

Five-sample protocol. For each question and draw, we select five of the ten indices without replacement. The seed is a hash of question identity, draw number, and a fixed protocol seed. We recompute the mode, confidence scores, vote statistics, and correctness. Formal clusters use only pair edges between selected indices, without importing paths through omitted traces. Signature generation sees the question only, and each translation sees one trace, so neither step requires the omitted samples.

Within a draw, the two budgets use the same questions with available scores. We report a paired difference only for models whose AUROC is defined at both budgets. This can change the eligible set relative to the main tables; on ZebraLogic it contains two or three models depending on the draw. Each metric is calculated before averaging over the 20 draws. The reported range describes draw sensitivity and is not a confidence interval. The comparison changes both the prediction and the uncertainty score, as a five-sample inference procedure would.

Benchmark	n/errors	Formal entropy	LLM entropy	Label entropy	Self-Brier
FOLIO	277/38	0.737	0.731	0.741	0.791
ZebraLogic	267/10	0.912	0.504	0.818	0.915
GSM-Hard	261/38	0.691	0.632	0.687	0.767

Table 9: Mean model-level error-detection AUROC on questions with ten accepted formalizations. The question set differs from the main analysis.

Benchmark	n/errors	Formal entropy	Formal + confidence	LLM entropy	LLM + confidence	Label entropy	Label + confidence
FOLIO	300/48	0.764	0.805	0.701	0.757	0.702	0.764
ZebraLogic	294/11	0.834	0.835	0.492	0.749	0.820	0.904
GSM-Hard	300/61	0.780	0.820	0.602	0.634	0.733	0.819

Table 10: Raw and confidence-tiebroken entropy on identical scored questions.

Benchmark	n/errors	Formal entropy	LLM entropy	LLM, later prompt
FOLIO	300/48	0.764	0.701	0.685
ZebraLogic	294/11	0.834	0.492	0.706

Table 11: Sensitivity to the later LLM equivalence prompt. Its confidence output is ignored for clustering; only the binary verdict is used. The stored run names establish lineage, but these later artifacts lack creation-time sample hashes.

Benchmark	B_s: 5	B_s: 10	5-sample range	H: 5	H: 10	5-sample range
OlympiadBench	0.727	0.721	[0.689, 0.782]	—	—	—
FOLIO	0.753	0.761	[0.722, 0.784]	0.732	0.764	[0.682, 0.763]
ZebraLogic	0.863	0.932	[0.700, 0.956]	0.794	0.865	[0.591, 0.857]
GSM-Hard	0.814	0.817	[0.789, 0.841]	0.757	0.780	[0.734, 0.785]

Table 12: Five- and ten-sample AUROC on paired sets within each draw. The ten-sample columns can differ from the main tables because score availability and eligible models depend on the draw. Ranges summarize five-sample results across draws.

D FORMALIZATION AND ARTIFACT INSPECTION

The three formal benchmarks have separate signature and translation templates. FOLIO and ZebraLogic share a repair template; the algebraic GSM-Hard protocol has a separate repair prompt that includes the sampled response and prohibits semantic correction.

Representations. For FOLIO and ZebraLogic, the auxiliary model translates the problem premises into Γ. FOLIO uses first-order predicates and quantifiers; each response's formula retains its verdict and claim about the conclusion, with optional intermediate claims stated in the response. ZebraLogic uses finite attributes, positions, and clue constraints; response formulas emphasize the answer and a small number of additional assignments. For GSM-Hard, the shared vocabulary contains only declarations for Real-valued quantities, with at most twelve quantities and no local helper declarations. Each ϕ_i supplies the adopted input values, equations, intermediate numerical claims, and final answer. GSM-Hard responses containing only an answer are encoded directly.

Symbols are compared by their generated names. FOLIO and ZebraLogic local helper names are not canonically aligned across traces; differences in names or retained commitments can split formulas even when the original arguments are similar.

Validity and repairs. Before comparing formulas, we check that each translation parses and is satisfiable with its shared background. FOLIO and ZebraLogic also require consistency with the sampled categorical answer and allow up to two repair calls per trace. Revised GSM-Hard translations must contain an explicit equality preserving the parsed rational final answer; this is checked before satisfiability. Up to two repairs can address syntax or answer preservation, but detected arithmetic contradictions are kept as recorded. Unsupported answers, empty translations, unresolved format errors, contradictory commitments, and solver timeouts remain separate singletons. If the shared signature cannot be obtained, the score is unavailable. Answer-only outputs can still produce weak, satisfiable formulas. Successful parsing and satisfiability do not verify preservation of the source argument.

Artifact inspection. We sampled 20 trace–formula pairs per formal benchmark. Fixed model quotas assign six, seven, and seven pairs; up to two failed translations are selected per model, with the remaining slots drawn from accepted translations. The fixed seed and full selected artifacts are recorded. This deliberately balanced sample cannot estimate population failure prevalence.

Claude Sonnet 4.5 inspected the complete question, stored trace, signature, and final formula under a fixed rubric. It labeled 33 pairs supported, 22 mismatched, and five unclear; three of the unclear labels result from invalid category responses. These are model assessments, not validated fidelity labels. In particular, several mismatch explanations objected only to omitted intermediate steps, despite the rubric allowing an abstraction of commitments. We retain the responses and do not report the 22 flags as 22 established translation errors.

Failures affect the partition, and accepted formulas may still retain much less than the original argument. An independent annotation study with an explicit abstraction policy would be needed to quantify translation fidelity.

E COST AND OPERATIONAL DETAILS

Benchmark	Valid traces	Failed	Calls/question	Solver median (s)	95th pct. (s)
FOLIO	2813/3000	187	12.31	0.133	0.194
ZebraLogic	2729/3000	271	13.53	0.200	1.981
GSM-Hard	2891/3000	109	7.71	0.119	0.163

Table 13: Formalization outcomes and solver-phase runtime. Calls/question equals one signature call plus recorded translation and repair attempts, averaged over all records, without response-cache reuse. Solver time excludes model inference, worker startup, and scheduling.

For FOLIO and ZebraLogic, ten-sample formalization takes 11 calls before repairs and at most 31 with two repairs per trace; the five-sample counts are six and 16. For revised GSM-Hard, let m be the number of explanation-bearing responses with a supported numerical answer. It uses one vocabulary call plus m translations, with at most two format repairs each, for a maximum of $1 + 3m$ calls. Syntactically answer-only responses are encoded directly and unsupported answers remain singletons. Vocabulary requests are identical across models for a shared question and can reuse the response cache. Pairwise equivalence requires at most $\binom{n}{2}$ comparisons, each with up to two implication checks. The solver can stop a comparison after finding a satisfiable direction. Failed traces skip their comparisons and remain separate singletons. We use a five-second timeout per direction. Worker failures are retried with process isolation; remaining unresolved comparisons would be recorded and left unmerged. None occurred in the final selected runs.

A timeout, parse error, or `unknown` result leaves a comparison unresolved and adds no equivalence edge. There is no LLM fallback. A finite solver budget can leave equivalent formulas in separate clusters.

Recorded calls and audit coverage. The call and cache counts describe different quantities:

- The 9,000 selected trace encodings contain 9,166 translation or repair attempts. Answer-only GSM-Hard encodings require none.
- Adding 900 per-record vocabulary requests gives 10,066 auxiliary calls before response-cache reuse. This includes reused historical translations, not just new calls during reanalysis.
- The primary artifact cache matches 2,326 unique requests, including 60 artifact assessments. This is the retained audit evidence, not total billed usage.

Unused pilot and pre-guard proposals are archived separately and excluded from the primary-pipeline count. Retained responses include token usage and repair reasons, but the cache is not a complete billing ledger. Original sampling-run totals combine several roles, so we report calls and solver time without claiming unmeasured dollar or end-to-end latency savings.

Each benchmark needs its own vocabulary, background representation, answer encoding, and translation policy. Prompt development used knowledge of these domains, but author time was not recorded; the study does not measure the cost of porting the method.

Alternative reasoning engines. The entropy calculation uses the resulting clusters and does not require a particular solver. Other SMT solvers, such as cvc5 (Barbosa et al., 2022), could check the equivalence conditions for formulas they support. A Lean implementation (de Moura & Ullrich, 2021) would require suitable formal encodings, validity checks, and proof search for both implications. We have not evaluated these alternatives. Different encodings and reasoning budgets can produce different clusters.

F PROMPTS AND WORKED CALCULATIONS

This section supplies the prompt templates for sampled answers and confidence, formal vocabularies, translations, repairs, and LLM equivalence judgments. Self-Brier and entropy calculation require no additional model prompt. The response templates were recovered from generation logs and checked against every selected question's recorded prompt hash. The formal templates and instantiated requests are included with their hashes in the artifacts. The LLM-equivalence templates have the provenance limits described in Appendix F.16. Listings preserve the instructions and examples, with typographic Unicode symbols transliterated to ASCII. The accompanying Markdown files retain the exact template text.

The historical FOLIO response prompt uses `Unknown`; some formalization templates call this verdict `Uncertain` and map it to the SMT constructor `unknown`. The response parser accepts `True`, `False`, and `Unknown`, case-insensitively. We preserve the historical template wording here.

F.1 WORKED SELF-BRIER CALCULATION

To isolate the association between confidence and agreement, keep the answers (A, A, A, B, B) fixed. Confidences $(0.90, 0.90, 0.90, 0.10, 0.10)$ give Self-Brier 0.01; reassigning those same values as $(0.10, 0.10, 0.90, 0.90, 0.90)$ gives 0.65. Both sets have mean confidence 0.58, vote uncertainty 0.40, and identical answer entropy. Only the association between confidence and answers changes. These illustrative calculations explain the score's behavior and make no claim about which modal answer is correct.

Figure 7 shows these two cases and an intermediate reassignment with confidences $(0.90, 0.90, 0.10, 0.90, 0.10)$, giving Self-Brier 0.33. Each tall bar is either a hesitant supporter or a confident dissenter.

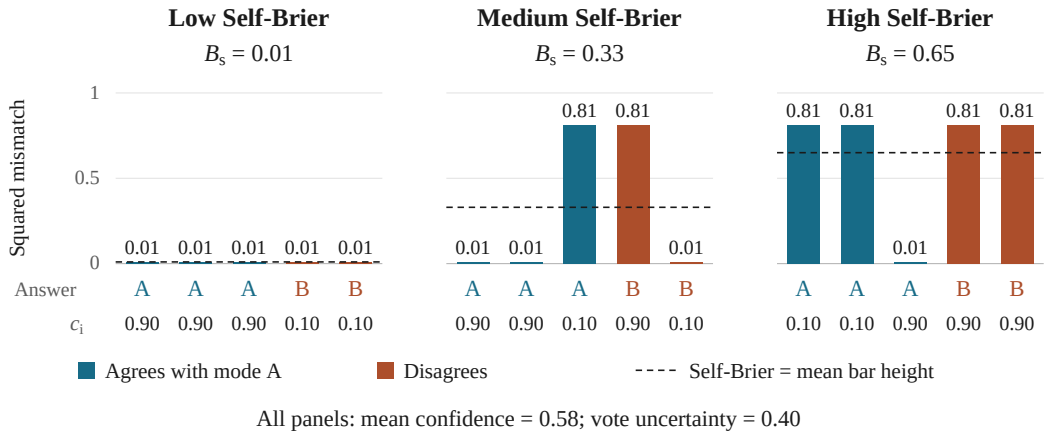


Figure 7: Self-Brier under reassignment of the same confidence values to fixed answers (A, A, A, B, B). Bars show per-sample squared mismatch $(c_i - y_i)^2$; dashed lines show their mean B_s. From left to right, high confidence shifts from supporters of the modal answer A to dissenters. Mean confidence, voting, and answer entropy stay fixed.

For a further calculation, consider the same five answers with reported confidences $(0.90, 0.70, 0.80, 0.95, 0.60)$. The mode is A, so agreement indicators are $(1, 1, 1, 0, 0)$. The score is

$$B_s = \frac{(0.90-1)^2 + (0.70-1)^2 + (0.80-1)^2 + 0.95^2 + 0.60^2}{5} = 0.2805.$$

The supporter contribution is $(0.01 + 0.09 + 0.04)/5 = 0.0280$ and the dissenter contribution is $(0.9025 + 0.36)/5 = 0.2525$. Mean-confidence uncertainty is $1 - 0.79 = 0.21$, and vote uncertainty is $1 - 3/5 = 0.40$. These quantities do not specify whether A is correct. Changing a reference answer leaves the score unchanged; it changes only the correctness label used in evaluation. The example is checked against the analysis implementation and is not an additional benchmark observation.

F.2 WORKED FORMAL-EQUIVALENCE CALCULATION

Figure 2 uses an evaluated FOLIO record. With the shared background Γ and aliases q for the query and g for the additional claim, the two True groups retain

$$\phi_1 = (v = \text{yes}) \wedge q, \qquad \phi_2 = (v = \text{yes}) \wedge q \wedge g.$$

The check $\Gamma \wedge \phi_2 \wedge \neg\phi_1$ is unsatisfiable, whereas $\Gamma \wedge \phi_1 \wedge \neg\phi_2$ is satisfiable. Thus one formula implies the other, but they are not equivalent and remain in separate clusters. The matrix in Section 3.2 extends these checks to all four clusters: rows supply the antecedent and columns the consequent. Its only off-diagonal `unsat` entry is $C_2 \to C_1$. Rechecking all 45 pairs of the ten returned judgments gives cluster sizes $(5, 2, 1, 2)$ and

$$H_{\text{formal}} = -0.5 \log_2 0.5 - 2(0.2 \log_2 0.2) - 0.1 \log_2 0.1 = 1.760964 \text{ bits}.$$

The verdicts alone occur with frequencies $(0.7, 0.1, 0.2)$, giving 1.156780 bits. The bundled record includes all ten original formulas and returned texts; the display's abbreviated formulas are checked against every member of their corresponding clusters.

F.3 ALGEBRAIC COMMITMENTS ON GSM-HARD

The revised translation preserves equations relating the quantities used by the sampled solution. A vocabulary is generated from the problem alone and reused across its responses. It contains declarations, not a solved answer. Each response's formula supplies its adopted input values, algebraic relationships, numerical intermediate claims, and final answer. A response that changes an implausible problem number retains that changed interpretation; the translator does not silently restore a preferred answer. If removing the answer and confidence tags leaves no text, the implementation directly constructs the exact numerical answer equality and makes no trace-translation call. This guard prevents the auxiliary model from supplying an unexpressed derivation.

For an illustrative problem with 20 units of blue fiber and half as much white fiber, a response that adds the two quantities and returns 30 supports:

```
(declare-const blue_fiber Real)
(declare-const white_fiber Real)
(declare-const answer Real)
(assert (= blue_fiber 20))
(assert (= white_fiber (/ blue_fiber 2)))
(assert (= answer (+ blue_fiber white_fiber)))
(assert (= answer 30))
```

Replacing the last assertion by `(assert (= answer 31))` while retaining the adopted equations produces an inconsistent commitment. It is recorded as such and becomes a singleton; the translator is not asked to correct the answer to 30 or delete an equation to achieve satisfiability. Syntax and exact-answer preservation can trigger up to two repair calls. The numerical answer must occur as an explicit equality before the satisfiability check, preventing an inconsistent formula from passing answer preservation by vacuous implication. This gate checks the translation, not agreement with a reference answer.

The earlier GSM-Hard encoding assigned final and intermediate numerical values without systematically retaining their algebraic relationships. Its results are retained as a matched ablation on the same sampled responses and scored questions. The revised comparison also changes the treatment of arithmetic contradictions: it preserves them rather than requesting a semantic repair. It therefore evaluates a revised translation protocol, not an isolated intervention on a single prompt sentence.

The new protocol was checked on three initial records before the full translation run. Inspection clarified how to preserve a response's adopted zero-clamping boundary case. A subsequent structural audit found that some model proposals added relationships to syntactically answer-only responses; the deterministic guard above enforces the stated abstraction policy. Those unused proposals and their requests remain archived. These decisions preceded the new comparative AUROC calculation. The reported primary formulas use the final policy throughout.

The model-averaged revised-minus-numerical difference is $+0.068$ with a 95% paired interval of $[+0.027, +0.112]$. Self-Brier, confidence, vote uncertainty, label entropy, the original LLM baseline,

Model	n/errors	Numerical	Revised	Δ [95% CI]	Failure AUROC
Sonnet 4.6	100/20	0.781	0.802	+0.021 [-0.037, +0.095]	0.670
Qwen3-235B	100/23	0.758	0.806	+0.048 [-0.026, +0.127]	0.741
GPT-OSS-20B	100/18	0.600	0.733	+0.134 [+0.056, +0.215]	0.544

Table 14: GSM-Hard error-detection AUROC on matched questions. Numerical denotes the previous value-commitment encoding; revised retains supported algebra and directly encodes answer-only outputs. Failure AUROC uses the fraction of rejected formalizations as the uncertainty score. Differences use paired question-bootstrap intervals.

and the scored prediction targets are unchanged in every GSM-Hard cell. GPT-OSS contributes 992 directly encoded numerical answers and eight unsupported answers; its gain does not demonstrate an advantage from richer algebraic reasoning. The individual intervals for Sonnet and Qwen include zero.

F.4 MATHEMATICS RESPONSE AND CONFIDENCE PROMPT

Recovered from the sampled-generation logs for Claude Sonnet 4.6, OlympiadBench: 100 questions; Qwen3-235B-A22B, OlympiadBench: 100 questions; GPT-OSS-20B, OlympiadBench: 100 questions; Claude Sonnet 4.6, GSM-Hard: 100 questions; Qwen3-235B-A22B, GSM-Hard: 100 questions; GPT-OSS-20B, GSM-Hard: 100 questions.

```
You are a mathematics problem solver. Solve the following problem, showing your reasoning step
↪  by step.

<problem>{criterion}</problem>

Think carefully. Algebra, arithmetic, and reasoning errors are the most common failure modes,
↪  so double-check each calculation before committing to an answer. If a sub-step produces a
↪  fraction, simplify it. If the problem asks for a specific form (reduced fraction, polar
↪  coordinates, etc.), match that form exactly.

Your final answer MUST be emitted inside the <judgment> tag using LaTeX (not plain text). Wrap
↪  the expression in \boxed{...} so the grader can locate it reliably. Keep the boxed content
↪  minimal: just the final numeric or symbolic answer, no extra prose, no units, no
↪  commentary.

Good examples:

- <judgment>\boxed{\frac{1}{2}}</judgment>
- <judgment>\boxed{(3, \frac{\pi}{2})}</judgment>
- <judgment>\boxed{42}</judgment>

End your response with EXACTLY these tags:

<judgment>\boxed{[final answer in LaTeX]}</judgment>
<confidence>[0-100, where 0 = completely uncertain, 100 = completely certain]</confidence>
```

F.5 FOLIO RESPONSE AND CONFIDENCE PROMPT

Recovered from the sampled-generation logs for Claude Sonnet 4.6, FOLIO: 100 questions; Qwen3-235B-A22B, FOLIO: 100 questions; GPT-OSS-20B, FOLIO: 100 questions.

```
You are a logic judge. Determine whether a conclusion logically follows from the given
↪  premises.

<premises>{document}</premises>

<conclusion_to_evaluate>{criterion}</conclusion_to_evaluate>

Think step by step through the logical argument. For each step, state which premise you use
↪  and what inference rule applies. Be explicit about what CAN and CANNOT be concluded.

Remember:
- "All A are B" does NOT mean "All non-A are non-B"
- "Some A are B" does NOT mean a specific individual is B
- If premises do not determine the conclusion either way, answer Unknown

Two worked examples:

Example 1:
```

```
Premises: Miroslav was a Czech choral conductor. Any choral conductor is a musician. Some
↪  musicians love music. Miroslav published a book in 1946.
Conclusion: A Czech published a book in 1946.
Analysis: Miroslav is Czech (premise 1). Miroslav published a book in 1946 (premise 4).
↪  Therefore someone Czech published a book in 1946 (existential introduction).
<judgment>True</judgment>
<confidence>95</confidence>

Example 2:
Same premises. Conclusion: Miroslav loved music.
Analysis: Miroslav is a choral conductor, therefore a musician (universal instantiation). Some
↪  musicians love music -- but "some" does NOT mean "all". We cannot conclude a specific
↪  individual loves music from an existential statement. Miroslav might or might not love
↪  music.
<judgment>Unknown</judgment>
<confidence>90</confidence>

Now evaluate the actual task. End your response with EXACTLY these tags:

<judgment>[True/False/Unknown]</judgment>
<confidence>[0-100, where 0 = completely uncertain, 100 = completely certain]</confidence>
```

F.6 ZEBRALOGIC REASONING-FIRST PROMPT

Recovered from the sampled-generation logs for Claude Sonnet 4.6, ZebraLogic: 100 questions; Qwen3-235B-A22B, ZebraLogic: 48 questions.

```
You are a logic puzzle solver. Each puzzle describes a small set of subjects (e.g. people,
↪  houses) and a set of attributes (e.g. color, drink, pet). Exactly one assignment of
↪  attributes to subjects is consistent with all the given clues. You are asked a single
↪  multiple-choice question about that unique assignment.

<puzzle>{document}
</puzzle>

<question>{criterion}</question>

Think step by step. Identify the subjects and attribute categories. Apply each clue as a
↪  constraint, using case analysis when a clue admits multiple possibilities, and eliminate
↪  branches that contradict later clues. Be explicit about which clue you are using at each
↪  step and why a branch is eliminated.

Do not guess. If the constraints genuinely leave more than one answer consistent -- i.e. the
↪  puzzle is underspecified as stated -- say so in your reasoning, but still pick the single
↪  letter you believe is most likely given the clues.

After your analysis, end your response with EXACTLY these tags:

<judgment>[single answer letter, e.g. A, B, C, D, or E]</judgment>
<confidence>[0-100, where 0 = completely uncertain, 100 = completely certain]</confidence>
```

F.7 ZEBRALOGIC INITIAL-ANSWER-COMMITMENT PROMPT

Recovered from the sampled-generation logs for Qwen3-235B-A22B, ZebraLogic: 52 questions; GPT-OSS-20B, ZebraLogic: 100 questions.

```
You are a logic puzzle solver. Each puzzle describes a small set of subjects (e.g. people,
↪  houses) and a set of attributes (e.g. color, drink, pet). Exactly one assignment of
↪  attributes to subjects is consistent with all the given clues. You are asked a single
↪  multiple-choice question about that unique assignment.

<puzzle>{document}
</puzzle>

<question>{criterion}</question>

**Answer commitment comes first.** Before any reasoning, write your best current guess in the
↪  final-answer tags below. You may revise during your analysis, but always leave a valid
↪  `<judgment>` tag in your output. If you run out of space before finishing the case
↪  analysis, the committed answer is what will be scored.

Required opening:

<judgment>[single answer letter]</judgment>
<confidence>[0-100]</confidence>
```

```
Then work through the puzzle step by step. Identify the subjects and attribute categories.
↪  Apply each clue as a constraint, using case analysis when a clue admits multiple
↪  possibilities, and eliminate branches that contradict later clues. Be explicit about which
↪  clue you are using at each step and why a branch is eliminated.

Do not guess. If the constraints genuinely leave more than one answer consistent -- i.e. the
↪  puzzle is underspecified as stated -- say so in your reasoning, but still commit to the
↪  single letter you believe is most likely given the clues.

If, after your full analysis, your answer changes from the initial commitment, end your
↪  response with an updated tag pair:

<judgment>[final answer letter]</judgment>
<confidence>[0-100, where 0 = completely uncertain, 100 = completely certain]</confidence>

The last `<judgment>` tag in your output is what gets scored.
```

F.8 FOLIO SHARED VOCABULARY AND PREMISES

```
You are a first-order-logic formaliser. Your job is to read a FOLIO problem (natural-language
↪  premises + a conclusion) and emit an **SMT-LIB 2 signature** encoding:

1. The individuals and sorts implicit in the premises.
2. The predicates and functions used in the premises.
3. All premises as SMT-LIB assertions.
4. A `Verdict` datatype (`yes` / `no` / `unknown` -- FOLIO uses `True`/`False`/`Uncertain`; map
↪  `Uncertain` to `unknown`).

Unlike ProofWriter, FOLIO premises are natural English prose. You must decide the signature
↪  yourself:

- Use a single uninterpreted sort `Entity` for the universe.
- Declare one unary predicate per adjective/attribute (`Student`, `Monkeypox`, `Mammal`, ...).
↪  Lowercase, replace spaces with underscores: `isStudent` -> `is_student`.
- Declare binary predicates for relations ("X eats Y", "X knows Y"). Arity follows the
↪  sentence structure.
- Declare individual constants for any named entities (people, specific diseases, etc.).
- Translate each premise to an SMT-LIB `(assert ...)`. Quantifiers:
  - "All X are Y" -> `(forall ((x Entity)) (=> (x_pred x) (y_pred x)))`.
  - "Some X is Y" -> `(exists ((x Entity)) (and (x_pred x) (y_pred x)))`.
  - "No X is Y" -> `(forall ((x Entity)) (=> (x_pred x) (not (y_pred x))))`.
  - "If ... then ..." -> `(=>  ...)`.
- Conjunctions / disjunctions inside a sentence: `(and ...)` / `(or ...)`.

**Crucial rules:**

- Every predicate must have range `Bool`.
- Preserve negation polarity precisely. "It is not the case that X" / "X is not Y" all get
↪  `(not ...)`.
- Use ASCII symbol names, lowercase with underscores.
- If a premise is ambiguous, pick the most natural FOL rendering and move on -- do not invent
↪  additional premises.
- Do NOT commit to a verdict; `answer` is the trace's job.

Respond with **only** the SMT-LIB text inside `<signature>...</signature>` tags.

---

### Example

PREMISES:
When the Monkeypox virus occurs in a being, it may get Monkeypox.
Monkeypox virus can occur in certain animals.
Humans are mammals.
Mammals are animals.
Symptoms of Monkeypox include fever, headache, muscle pains, and tiredness.
People feel tired when they get the flu.

CONCLUSION:
There is an animal.

<signature>
(declare-sort Entity 0)
(declare-const monkeypox_virus Entity)
(declare-const flu Entity)
(declare-fun animal (Entity) Bool)
(declare-fun mammal (Entity) Bool)
(declare-fun human (Entity) Bool)
(declare-fun occur_in (Entity Entity) Bool)
(declare-fun get (Entity Entity) Bool)
(declare-fun feels_tired (Entity) Bool)
```

```
(declare-datatypes () ((Verdict (yes) (no) (unknown))))
(declare-const answer Verdict)
; "When the Monkeypox virus occurs in a being, it may get Monkeypox."
; (Existential: there is some being in which the virus occurs and gets it.)
(assert (exists ((x Entity))
          (and (occur_in monkeypox_virus x) (get x monkeypox_virus))))
; "Monkeypox virus can occur in certain animals."
(assert (exists ((x Entity))
          (and (animal x) (occur_in monkeypox_virus x))))
; "Humans are mammals."
(assert (forall ((x Entity)) (=> (human x) (mammal x))))
; "Mammals are animals."
(assert (forall ((x Entity)) (=> (mammal x) (animal x))))
; "People feel tired when they get the flu."
(assert (forall ((x Entity))
          (=> (and (human x) (get x flu)) (feels_tired x))))
; Symptoms premise deliberately omitted -- ambiguous symptom-of relation
; would over-complicate the signature; the trace does not rely on it
; for the conclusion.
</signature>

---

Now produce the signature for the following instance:

PREMISES:
{document}

CONCLUSION:
{criterion}
```

F.9 FOLIO TRACE COMMITMENTS

```
You are a first-order-logic formaliser. You have already been given:
- A FOLIO problem (natural-language premises + conclusion).
- An **SMT-LIB signature** containing all declared predicates and all premise assertions, plus
↪  a `Verdict` enum (`yes | no | unknown`) and `answer : Verdict`.
- A **reasoning trace** from a judge who attempted the problem.
- The judge's final committed **verdict**: `True` / `False` / `Uncertain`.

Your job: emit SMT-LIB `(assert ...)` forms that encode the trace's *commitment* on top of the
↪  signature.

**What to assert:**

1. **The verdict mapping.** The trace's stated verdict maps to a constructor of `Verdict`:
   - `True`       -> `yes`
   - `False`      -> `no`
   - `Uncertain`  -> `unknown`
   Emit `(assert (= answer <constructor>))`.

2. **The trace's claim about the conclusion `phi`.**
   - `True`: the trace claims `phi` follows from the premises. Emit `(assert phi)`.
   - `False`: the trace claims `not phi` follows. Emit `(assert (not phi))`.
   - `Uncertain`: the trace claims neither follows -- do NOT assert `phi` or `not phi`. The
   ↪  verdict mapping alone is enough.

3. **Intermediate lemmas the trace explicitly uses** (optional). Include only if directly
↪  stated in the trace and expressible in the signature.

**Rules:**

- Use only symbols declared in the signature. Do not redeclare anything.
- Do not re-emit any of the premise assertions from the signature.
- Quantify correctly. A conclusion like "There is an animal" becomes `(exists ((x Entity))
↪  (animal x))` -- not a bare atom.
- Keep intermediate lemmas conservative: only what the trace names explicitly.

Emit only the `(assert ...)` forms inside `<formalization>...</formalization>` tags.

---

### Example -- True verdict

SIGNATURE (abbreviated):
(declare-sort Entity 0)
(declare-fun animal (Entity) Bool)
(declare-fun mammal (Entity) Bool)
(declare-fun human (Entity) Bool)
(declare-datatypes () ((Verdict (yes) (no) (unknown))))
(declare-const answer Verdict)
... (premises present but omitted here)
```

```
CONCLUSION:
There is an animal.

TRACE:
From "Monkeypox virus can occur in certain animals" we have an existential witness -- there
↪  exists an entity that is an animal. So the conclusion "There is an animal" follows
↪  directly.

FINAL ANSWER: True

<formalization>
(assert (exists ((x Entity)) (animal x)))
(assert (= answer yes))
</formalization>

---

### Example -- Uncertain verdict

CONCLUSION:
Every mammal feels tired.

TRACE:
The premises say humans feel tired when they get the flu, and humans are mammals. But the
↪  premises don't establish that every mammal feels tired -- non-human mammals might not. So
↪  we cannot conclude the universal, and also cannot disprove it. Uncertain.

FINAL ANSWER: Uncertain

<formalization>
; Uncertain: no assertion on the conclusion -- only the verdict mapping.
(assert (= answer unknown))
</formalization>

---

Now produce the formalization for the following trace:

SIGNATURE:
{signature}

CONCLUSION:
{criterion}

TRACE (the judge's reasoning):
{trace}

FINAL ANSWER (the verdict the trace committed to):
{answer_letter}
```

F.10 ZebraLogic shared vocabulary and clues

```
You are a logic-puzzle formaliser. Your job is to read a ZebraLogic puzzle and emit an
↪  **SMT-LIB 2 signature** that captures:

1. The puzzle's vocabulary (sorts, functions, constants), and
2. The puzzle's constraints (the `distinct` axioms AND the numbered clues).

The resulting signature is the *puzzle-wide common ground*: all 10 reasoning traces share it.
↪  Each trace will later add its own assertions pinning down the assignment it committed to.

A ZebraLogic puzzle has:
- A fixed number of positions (houses), numbered `1` through `N`.
- Several attribute *categories* (e.g. `Name`, `Color`, `Drink`), each with exactly `N`
↪  distinct *values*.
- A set of numbered clues that further constrain which assignment is valid.

Your signature must:
1. Declare a `House` datatype enumerating the house constants `h1, h2, ..., hN`.
2. Declare one enumerated datatype per attribute category, listing every value (lowercase,
↪  underscores for spaces/punctuation).
3. Declare a total function `<category>(House) -> <Category>` for each category, with a
↪  lowercase function name. For example: category `Name` -> `(declare-fun name (House)
↪  Name)`.
4. Declare a constant `answer` of type `Letter` -- an enumerated datatype with constructors
↪  matching the multiple-choice letters (`A B C D E` etc.).
5. Emit one `distinct` axiom per category: each category's function must map the `N` houses to
↪  `N` distinct values.
6. Emit one `(assert ...)` per numbered clue, translating each clue to SMT-LIB. See the
↪  reference below.
```

```
**Value-name normalisation:** lowercase, replace spaces/hyphens/punctuation with `_`. Example:
↪  `"iPhone 13"` -> `iphone_13`, `"Ford F-150"` -> `ford_f_150`, `"pop-music"` -> `pop_music`.

**Clue translation reference** (idioms to recognise; reuse verbatim when applicable):

- "X is in the k-th house" -> `(= (<cat> hK) X)` where `<cat>` is the relevant category.
- "X is to the left of Y" -> `(exists ((i House) (j House)) (and (= (<cat1> i) X) (= (<cat2>
↪  j) Y) (< (pos i) (pos j))))`. Use a helper function `(declare-fun pos (House) Int)` with
↪  `(assert (= (pos h1) 1)) ... (assert (= (pos hN) N))` if any ordering clues appear.
- "X is directly left of Y" -> same but `(= (+ (pos i) 1) (pos j))`.
- "The person who has X also has Y" (attribute co-location) -> `(forall ((h House)) (=> (=
↪  (<catX> h) X) (= (<catY> h) Y)))`.
- "X and Y are in different houses" (rarely needed -- already implied by `distinct` when X,Y
↪  are same category).
- "X is not in the first house" -> `(not (= (<cat> h1) X))`.

**Crucial rules:**
- One lowercase function per category. ASCII only.
- Each `distinct` axiom lists ALL `N` houses in order: `(assert (distinct (name h1) (name h2)
↪  (name h3)))`.
- For ordering clues, declare `pos` and pin `(= (pos h1) 1)` ... `(= (pos hN) N)` exactly once.
- If a clue is ambiguous or you cannot translate it confidently, skip it (better to
↪  under-constrain than to corrupt the signature). Never invent clues.
- Do NOT assert anything about `answer` -- that's the trace formaliser's job.

Respond with **only** the SMT-LIB text inside `<signature>...</signature>` tags. No prose
↪  outside.

---

### Example 1 (2 houses, 2 categories, 2 clues, no ordering)

PUZZLE:
There are 2 houses, numbered 1 to 2 from left to right, as seen from across the street. Each
↪  house is occupied by a different person. Each house has a unique attribute for each of the
↪  following characteristics:
 - Each person has a unique name: `Eric`, `Arnold`
 - People have unique hair colors: `brown`, `black`

## Clues:
1. Arnold is in the first house.
2. The person who has brown hair is not in the first house.

QUESTION:
What is Name of the person who lives in House 2?

CHOICES:
A. Arnold
B. Eric

<signature>
(declare-datatypes () ((House (h1) (h2))))
(declare-datatypes () ((Name (eric) (arnold))))
(declare-datatypes () ((HairColor (brown) (black))))
(declare-datatypes () ((Letter (A) (B))))
(declare-fun name (House) Name)
(declare-fun haircolor (House) HairColor)
(declare-const answer Letter)
(assert (distinct (name h1) (name h2)))
(assert (distinct (haircolor h1) (haircolor h2)))
; Clue 1: Arnold is in the first house.
(assert (= (name h1) arnold))
; Clue 2: The person who has brown hair is not in the first house.
(assert (not (= (haircolor h1) brown)))
</signature>

---

### Example 2 (3 houses, 3 categories, ordering + co-location clues)

PUZZLE:
There are 3 houses, numbered 1 to 3 from left to right, as seen from across the street. Each
↪  house is occupied by a different person. Each house has a unique attribute for each of the
↪  following characteristics:
 - Each person has a unique name: `Eric`, `Arnold`, `Peter`
 - Everyone has a favorite smoothie: `desert`, `watermelon`, `cherry`
 - People have unique favorite book genres: `science fiction`, `romance`, `mystery`

## Clues:
1. The person who likes Cherry smoothies is somewhere to the left of the person who loves
↪  mystery books.
2. Arnold is the person who loves mystery books.
```

```
3. The person who loves science fiction books is not in the first house.
4. The Desert smoothie lover is directly left of the person who loves mystery books.
5. Peter is in the first house.

QUESTION:
What is Smoothie of the person who lives in House 3?

CHOICES:
A. cherry
B. desert
C. watermelon

<signature>
(declare-datatypes () ((House (h1) (h2) (h3))))
(declare-datatypes () ((Name (eric) (arnold) (peter))))
(declare-datatypes () ((Smoothie (desert) (watermelon) (cherry))))
(declare-datatypes () ((BookGenre (science_fiction) (romance) (mystery))))
(declare-datatypes () ((Letter (A) (B) (C))))
(declare-fun name (House) Name)
(declare-fun smoothie (House) Smoothie)
(declare-fun bookgenre (House) BookGenre)
(declare-fun pos (House) Int)
(declare-const answer Letter)
(assert (= (pos h1) 1))
(assert (= (pos h2) 2))
(assert (= (pos h3) 3))
(assert (distinct (name h1) (name h2) (name h3)))
(assert (distinct (smoothie h1) (smoothie h2) (smoothie h3)))
(assert (distinct (bookgenre h1) (bookgenre h2) (bookgenre h3)))
; Clue 1: Cherry smoothie is to the left of the mystery-books person.
(assert (exists ((i House) (j House))
          (and (= (smoothie i) cherry)
               (= (bookgenre j) mystery)
               (< (pos i) (pos j)))))
; Clue 2: Arnold loves mystery books.
(assert (forall ((h House))
          (=> (= (name h) arnold) (= (bookgenre h) mystery))))
; Clue 3: The science-fiction reader is not in the first house.
(assert (not (= (bookgenre h1) science_fiction)))
; Clue 4: Desert-smoothie person is directly left of the mystery-books person.
(assert (exists ((i House) (j House))
          (and (= (smoothie i) desert)
               (= (bookgenre j) mystery)
               (= (+ (pos i) 1) (pos j)))))
; Clue 5: Peter is in the first house.
(assert (= (name h1) peter))
</signature>

---

Now produce the signature for the following puzzle:

PUZZLE:
{document}

QUESTION:
{criterion}
```

F.11 ZEBRALOGIC TRACE COMMITMENTS

```
You are a logic-puzzle formaliser. You have already been given:
- A ZebraLogic puzzle,
- An **SMT-LIB signature** (sorts, functions, constants, `distinct` axioms, and the puzzle's
↪  clue constraints),
- A **reasoning trace** produced by a judge who attempted the puzzle,
- The judge's final **committed answer letter** (A, B, C, ...).

Your job: emit the **minimal formal commitment** the trace makes, as SMT-LIB 2 `(assert ...)`
↪  statements on top of the already-loaded signature.

**Minimality is the goal.** The signature already encodes the puzzle's clue constraints; your
↪  job is to pin down *what the trace claims the answer is*, not to re-derive every cell of
↪  the grid. Every extra assertion is a chance for the formalization to become inconsistent
↪  with the clues (if the trace's intermediate reasoning was subtly wrong) and get rejected
↪  as `unsat`. Keep the formalization short.

**What to assert (in order of priority; stop once you have a complete answer commitment):**

1. **The answer mapping.** Every trace ends with a letter answer. This is mandatory:
   - `(assert (= answer <Letter>))` -- pin the letter.
```

```
   - `(assert (= (<cat> hK) v))` -- pin the single attribute value that the letter corresponds
   ↪  to in the question (e.g. if the question asks "What is the smoothie of House 3?" and
   ↪  the letter maps to "watermelon", assert `(= (smoothie h3) watermelon)`).
   These two lines are sufficient for the answer commitment and are the ONLY assertions most
   ↪  traces need.

2. **Witness constraints (OPTIONAL, include only if required).** Only add further assertions
↪  when the answer mapping alone is not enough to distinguish the trace from another trace
↪  that reached a different answer. Examples:
   - A trace that says "Peter is in House 1" as the linchpin of its reasoning -- you MAY
   ↪  assert `(= (name h1) peter)`.
   - A trace that reaches a different intermediate conclusion than a sibling trace but the
   ↪  same final answer -- assert ONE differentiating witness so the partition can separate
   ↪  them.

3. **Do NOT assert the full grid.** Do NOT transcribe every name, smoothie, book-genre, and
↪  colour the trace walks through. Each of those is an opportunity for the formalization to
↪  become inconsistent with a clue if the trace was subtly wrong. Stick to step 1; add step 2
↪  sparingly.

**Rules:**

- Use only symbols declared in the signature. If a value name in the trace doesn't match any
↪  constructor in the signature, it is a trace/signature mismatch -- skip that assertion
↪  rather than invent a new constructor.
- Normalise value names exactly as in the signature (lowercase, underscores).
- Do NOT redeclare sorts or functions. Do NOT re-assert `distinct` or any clue.
- If the trace is ambiguous (e.g. it computes several candidate assignments and doesn't pick
↪  one), assert only the final committed letter plus the one attribute it maps to. If even
↪  the letter is unclear, assert only `(= answer <Letter>)` using the stated `FINAL ANSWER`
↪  value.
- Emit only the `(assert ...)` forms inside `<formalization>...</formalization>` tags. No
↪  prose outside.

---

### Example 1 (2x2 puzzle: minimal answer-only formalization)

SIGNATURE:
(declare-datatypes () ((House (h1) (h2))))
(declare-datatypes () ((Name (eric) (arnold))))
(declare-datatypes () ((HairColor (brown) (black))))
(declare-datatypes () ((Letter (A) (B))))
(declare-fun name (House) Name)
(declare-fun haircolor (House) HairColor)
(declare-const answer Letter)
(assert (distinct (name h1) (name h2)))
(assert (distinct (haircolor h1) (haircolor h2)))
(assert (= (name h1) arnold))
(assert (not (= (haircolor h1) brown)))

QUESTION:
What is Name of the person who lives in House 2?

CHOICES:
A. Arnold
B. Eric

TRACE:
Clue 1 says Arnold is in House 1, so House 2 must be Eric. Clue 2 says
brown hair is not in House 1, so House 1 is black-haired and House 2
is brown-haired.

Therefore:
- House 1: Arnold, black hair
- House 2: Eric, brown hair

The question asks for the name in House 2. That is Eric, which is
choice B.

FINAL ANSWER: B

<formalization>
; Minimal commitment: the letter and the single attribute it maps to.
(assert (= answer B))
(assert (= (name h2) eric))
</formalization>

---

### Example 2 (3x3 puzzle: answer + one witness)

SIGNATURE:
```

```
(declare-datatypes () ((House (h1) (h2) (h3))))
(declare-datatypes () ((Name (eric) (arnold) (peter))))
(declare-datatypes () ((Smoothie (desert) (watermelon) (cherry))))
(declare-datatypes () ((BookGenre (science_fiction) (romance) (mystery))))
(declare-datatypes () ((Letter (A) (B) (C))))
(declare-fun name (House) Name)
(declare-fun smoothie (House) Smoothie)
(declare-fun bookgenre (House) BookGenre)
(declare-const answer Letter)
... (clue assertions omitted for brevity)

QUESTION:
What is Smoothie of the person who lives in House 3?

CHOICES:
A. cherry
B. desert
C. watermelon

TRACE:
From Clue 5, Peter is in House 1. From Clue 2, Arnold loves mystery.
Clue 4 says desert-smoothie is directly left of mystery, so mystery
is in House 2 or House 3. Clue 1 says cherry is to the left of mystery,
so mystery can't be in House 1 anyway. If mystery is in House 2, then
desert is in House 1 and cherry must be to the left of House 2, which
requires cherry in House 1 -- but desert is already there. So mystery
is in House 3, desert in House 2, cherry in House 1.

Final smoothies: House 1 = cherry, House 2 = desert, House 3 = watermelon.

The question asks for the smoothie in House 3. That's watermelon,
which is choice C.

FINAL ANSWER: C

<formalization>
; Minimal commitment: the letter and the single attribute it maps to.
; (Peter/Arnold/mystery intermediates are NOT encoded --- they are not
; needed to differentiate this trace's answer from a wrong one.)
(assert (= answer C))
(assert (= (smoothie h3) watermelon))
</formalization>

---

Now produce the formalization for the following trace:

SIGNATURE:
{signature}

QUESTION:
{criterion}

TRACE (the judge's reasoning):
{trace}

FINAL ANSWER (the letter the trace committed to):
{answer_letter}
```

F.12 FOLIO AND ZEBRALOGIC REPAIR

```
You previously emitted an SMT-LIB 2 formalization that Z3 rejected. Read the error and emit a
↪  corrected version.

**Error verbatim from Z3:**

{error}

**What you previously emitted:**

{previous_formalization}

**Context -- SIGNATURE (already parsed; do NOT redeclare any of these):**

{signature}

**Repair guidance** (pick the one that matches the error):

- **Parse error / unexpected token:** check parenthesis balance. Every `(declare-...)` and
↪  `(assert ...)` must have matching parens. `(and p q)`, not `(and p q`.
```

```
- **Unknown / undeclared symbol:** the symbol you used is not in the signature. Either it's misspelled (compare case and underscores carefully -- `iPhone 13` was normalised to `iphone_13`), or the trace mentioned a value the puzzle does not list. If the trace mentions a value that is not in the signature, do NOT invent it -- skip that assertion.

- **Sort mismatch / type error:** a function was applied to a value of the wrong sort. Re-check: `(color h1)` returns a `Color`, not a `House`. Do not compare values across different enumerated sorts.

- **Unsat (formalization has no models):** your assertions are internally contradictory. Common causes: you asserted the same attribute twice with conflicting values, or you asserted a value that violates one of the `distinct` axioms. Revisit what the trace ACTUALLY commits to -- only include assignments the trace itself states explicitly and consistently.

- **Formalization + stated answer is unsat:** the trace says the answer is one letter but your formalization forces a different letter. Re-read the trace's final-answer block carefully. The `(assert (= answer <letter>))` must match the letter the trace actually committed to.

Emit the corrected SMT-LIB inside `<formalization>...</formalization>` tags. No prose outside.
```

F.13 GSM-HARD ALGEBRAIC VOCABULARY

```
You construct a shared SMT-LIB 2 vocabulary for algebraic commitments in solutions to a word problem.

Read only the problem. Declare answer as Real and declare the named input and intermediate quantities needed to express its arithmetic relationships. Use at most twelve Real constants in total, with short, descriptive lowercase names. Add a short SMT comment explaining each quantity and its unit. Use the same name for the same quantity throughout. Prefer quantities such as blue_fiber, white_fiber, hourly_rate, regular_hours, overtime_hours, subtotal, and answer over step1 or x.

Output declarations and comments only, inside <signature>...</signature>. Do not assert values, solve the problem, or impose equations here. The separate translation of each sampled solution will assert the input values and algebraic relationships that the solution commits to. Keeping these assertions out of the shared vocabulary allows different sampled interpretations to remain distinguishable.

Use no functions, quantifiers, datatypes, definitions, or fresh sorts. All declared quantities are Real, even when a count is normally integral. Do not impose unstated positivity, integrality, or feasibility conditions. Large or implausible numbers must not be silently replaced.

Example problem:
A robe takes 20 units of blue fiber and half as much white fiber. How many units in total?

<signature>
(declare-const blue_fiber Real) ; units of blue fiber
(declare-const white_fiber Real) ; units of white fiber
(declare-const answer Real) ; total units of fiber
</signature>

PROBLEM:
{problem}
```

F.14 GSM-HARD ALGEBRAIC COMMITMENTS

```
Translate one sampled solution into its algebraic commitments in SMT-LIB 2. You receive the problem, a shared vocabulary, the sampled response, and its extracted final answer. You are a translator, not a solver or a corrector.

Encode arithmetic relationships, not just the numerical results of intermediate steps. If the response uses "white fiber is half of blue fiber" and "the total is their sum", retain equations such as (= white_fiber (/ blue_fiber 2)) and (= answer (+ blue_fiber white_fiber)). Bind input quantities to the values used in that response. The problem can resolve a reference such as "half as much" or supply an operand for a calculation the response explicitly endorses.

Preserve the sampled interpretation, including an explicit change to an implausible problem number. Do not substitute the interpretation you believe is correct. Omit abandoned alternatives and calculations the response explicitly retracts. Retain the final adopted equations and explicitly committed intermediate numerical results. Assert the extracted final answer exactly as stated, even if it contradicts those equations. An inconsistent translation is an informative artifact; never change a number or delete a supported equation to make it satisfiable.
```

```
Preserve adopted boundary cases. If the response says that a negative remainder means zero
↪  items can be sold, its saleable remainder is clamped at zero: encode an ite or the
↪  explicitly selected zero branch. Do not also assert an unconditional subtraction for that
↪  saleable quantity. Similarly, retain an explicitly adopted rounding, cap, or discrete case
↪  rather than inventing an incompatible continuous equation. If an operation cannot be
↪  expressed using the shared vocabulary, retain the supported partial constraints and
↪  numerical answer.

Do not invent an unexpressed derivation. A response containing only a final answer supports an
↪  answer equality, not a reconstructed solution. Do not infer all problem equations solely
↪  from the final number. In comments before the assertions, identify short quotations or
↪  explicit calculations supporting each retained relationship.

Use only declared constants and Real arithmetic (+, -, *, /), equalities, inequalities, and
↪  Boolean combinations needed by the sampled argument. For discrete choices, encode the
↪  particular case the response adopts; do not invent a search over alternatives. Do not add
↪  new declarations, definitions, quantifiers, or assertions about confidence. Use exact
↪  integers and rational expressions. Write negative numbers as (- 3). Do not round, evaluate
↪  away the structure of a calculation, or introduce division by zero.

Examples use the shared constants blue_fiber, white_fiber, answer.

RESPONSE: Blue fiber is 20. White fiber is half of that, 10. The total is 20 + 10 = 30.
FINAL ANSWER: 30
<formalization>
; "White fiber is half of that"; "The total is 20 + 10"
(assert (= blue_fiber 20))
(assert (= white_fiber (/ blue_fiber 2)))
(assert (= white_fiber 10))
(assert (= answer (+ blue_fiber white_fiber)))
(assert (= answer 30))
</formalization>

RESPONSE: Blue fiber is 20. White fiber is half of that, 10. Adding them gives 31.
FINAL ANSWER: 31
<formalization>
; Retain the arithmetic mistake; do not repair the answer to 30.
(assert (= blue_fiber 20))
(assert (= white_fiber (/ blue_fiber 2)))
(assert (= white_fiber 10))
(assert (= answer (+ blue_fiber white_fiber)))
(assert (= answer 31))
</formalization>

RESPONSE: <judgment>30</judgment><confidence>90</confidence>
FINAL ANSWER: 30
<formalization>
(assert (= answer 30))
</formalization>

Emit only <formalization>...</formalization>, containing SMT comments and (assert ...)
↪  commands. If the response has no interpretable mathematical commitment, emit ;
↪  no_commitment inside those tags.

PROBLEM:
{problem}

SHARED VOCABULARY:
{signature}

SAMPLED RESPONSE:
{trace}

EXTRACTED FINAL ANSWER:
{answer}
```

F.15 GSM-HARD SYNTAX AND ANSWER-PRESERVATION REPAIR

```
Repair only the syntax or vocabulary violation in the proposed SMT-LIB translation below.
↪  Preserve the sampled response's adopted algebraic relationships, intermediate numerical
↪  claims, and final answer. Do not solve the problem, correct arithmetic, or delete a
↪  supported commitment merely to obtain satisfiability. Do not invent a derivation for an
↪  answer-only response. Use only the shared Real constants and emit only (assert ...)
↪  commands and comments inside <formalization>...</formalization>.

PROBLEM:
{problem}

SHARED VOCABULARY:
{signature}
```

```
SAMPLED RESPONSE:
{trace}

EXTRACTED FINAL ANSWER:
{answer}

PREVIOUS TRANSLATION:
{previous}

DIAGNOSTIC:
{diagnostic}
```

F.16 LLM-EQUIVALENCE PROMPT PROVENANCE

These archived templates match the hashes in the reanalysis manifest. The dataset-based prompt loader selects the generic binary template for OlympiadBench and GSM-Hard and separate templates for FOLIO and ZebraLogic. Individual binary-oracle responses were not retained, so these template hashes do not establish a complete audit of the historical calls. The later FOLIO and ZebraLogic runs also lack creation-time sample hashes. Their requested confidence is ignored for clustering; only the binary verdict is used.

F.17 BINARY LLM EQUIVALENCE: MATHEMATICS

Response limit: 256 tokens.

```
Compare two judge reasoning traces for a general categorical-judgment task.

<reasoning_a>{reasoning_a}</reasoning_a>

<reasoning_b>{reasoning_b}</reasoning_b>

Determine whether these two traces reach the same conclusion via the same derivation -- NOT
↪  merely whether the conclusion labels match.

STRICT equivalence rules:
- DIFFERENT final conclusion -> NOT_EQUIVALENT, regardless of reasoning similarity.
- Same conclusion but different load-bearing evidence -> NOT_EQUIVALENT. Traces must cite the
↪  same specific evidence (not just any evidence that supports the conclusion).
- Same conclusion but different inference structure -> NOT_EQUIVALENT. The key derivation step
↪  -- the one that actually determines the conclusion -- must be the same in both traces.
- Restatements, paraphrases, and stylistic variations of the same content ARE equivalent.
↪  Match on derivation content, not on surface form.

Default toward NOT_EQUIVALENT when uncertain -- a false merge is far worse than a false split
↪  for semantic-entropy clustering.

<verdict>[EQUIVALENT or NOT_EQUIVALENT]</verdict>
```

F.18 BINARY LLM EQUIVALENCE: FOLIO

Response limit: 256 tokens.

```
Compare two judge reasoning traces for a first-order logic entailment task. The judge decides
↪  whether a conclusion follows from a set of premises; labels are True, False, or Unknown.

<reasoning_a>{reasoning_a}</reasoning_a>

<reasoning_b>{reasoning_b}</reasoning_b>

Determine whether these two traces reach the same verdict via the same logical derivation --
↪  NOT merely whether they reach the same verdict.

STRICT equivalence rules:
- DIFFERENT final verdict (True / False / Unknown) -> NOT_EQUIVALENT, no matter how similar
↪  the reasoning is.
- Same verdict but different derivation -> NOT_EQUIVALENT. Specifically, the traces must agree
↪  on ALL of:
  * The SAME subset of premises cited as load-bearing -- the premises without which the
  ↪  verdict could not be reached. If trace A uses premises 1, 3, 5 and trace B uses premises
  ↪  1, 4, 5, they are NOT equivalent even if both conclude True.
  * The SAME inference rules applied at each load-bearing step (modus ponens, modus tollens,
  ↪  universal instantiation, existential introduction, contrapositive, etc.). A trace
  ↪  deriving via universal instantiation and one deriving the same fact via existential
  ↪  elimination are NOT equivalent.
```

```
  * The SAME key step -- the inference that actually pins down the verdict. Traces that happen
  ↪  to reach the same verdict via different key steps (e.g. one uses a proof-by-cases,
  ↪  another uses direct derivation) are NOT equivalent.
  * For "Unknown" verdicts: the SAME reason the conclusion cannot be determined. "Some doesn't
  ↪  entail specific individual" is NOT equivalent to "missing premise linking A to B" --
  ↪  these are different logical gaps.
- Trivial relabelings ARE equivalent -- "premise 3" and "the third premise" refer to the same
↪  content. "Modus ponens" and "MP" are the same rule.
- Stylistic differences (preamble, bullet formatting, summary sentences, restatement of
↪  premises in logical notation) are IRRELEVANT. Match on the derivation content.
- A trace truncated mid-derivation must be compared only on the derivation actually performed.
↪  A truncated trace that reaches the same verdict as a complete trace by accident is NOT
↪  equivalent.

Default toward NOT_EQUIVALENT when uncertain -- a false merge is far worse than a false split
↪  for semantic-entropy clustering.

<verdict>[EQUIVALENT or NOT_EQUIVALENT]</verdict>
```

F.19 BINARY LLM EQUIVALENCE: ZEBRALOGIC

Response limit: 256 tokens.

```
Compare two judge reasoning traces for a logic-grid puzzle (zebra-puzzle / constraint
↪  satisfaction task). The puzzle has a unique valid assignment, and the judge has picked one
↪  letter.

<reasoning_a>{reasoning_a}</reasoning_a>

<reasoning_b>{reasoning_b}</reasoning_b>

Determine whether these two traces reach the same answer via the same constraint-propagation
↪  path -- NOT merely whether they pick the same letter.

STRICT equivalence rules:
- DIFFERENT final answer letter -> NOT_EQUIVALENT, no matter how similar the reasoning is.
- Same letter but different constraint derivation -> NOT_EQUIVALENT. Specifically, the traces
↪  must agree on:
  * The SAME subset of clues used as load-bearing (the clues without which the answer could
  ↪  not be pinned down).
  * The SAME order of constraint propagation, up to reorderings that are provably equivalent.
  ↪  If one trace eliminates option X by clue 3 then clue 7, and the other eliminates X by
  ↪  clue 9 alone, those are NOT equivalent derivations.
  * The SAME case split structure. If trace A performs a 2-way case split on "house 2 is red
  ↪  or blue" and trace B does not split at all, they are NOT equivalent -- one of them is
  ↪  implicitly appealing to a clue the other isn't.
  * The SAME treatment of each contradicting branch. If both traces must eliminate a branch,
  ↪  they must eliminate it using the same constraint.
- Stylistic similarities (preamble, bullet formatting, summary sentences) are IRRELEVANT. Two
↪  traces with identical opening paragraphs but different clue citations are NOT equivalent.
- Trivial relabelings (e.g. "clue 3" vs "the third clue") ARE equivalent -- match on content,
↪  not citation format.
- If a trace is truncated mid-derivation, compare only the derivation that was actually
↪  performed. A truncated trace that reaches the same letter by accident is NOT equivalent to
↪  a complete trace that derives that letter.

Default toward NOT_EQUIVALENT when uncertain -- a false merge is far worse than a false split
↪  for semantic-entropy clustering.

<verdict>[EQUIVALENT or NOT_EQUIVALENT]</verdict>
```

F.20 LATER LLM EQUIVALENCE PROMPT: FOLIO

Response limit: 4,096 tokens.

```
Compare two judge reasoning traces for a first-order logic entailment task. The judge decides
↪  whether a conclusion follows from a set of premises; labels are True, False, or Unknown.

<reasoning_a>{reasoning_a}</reasoning_a>

<reasoning_b>{reasoning_b}</reasoning_b>

Determine whether these two traces reach the same verdict via the same logical derivation --
↪  NOT merely whether they reach the same verdict.

STRICT equivalence rules:
- DIFFERENT final verdict (True / False / Unknown) -> NOT_EQUIVALENT, no matter how similar
↪  the reasoning is.
```

```
- Same verdict but different derivation -> NOT_EQUIVALENT. Specifically, the traces must agree on ALL of:
  * The SAME subset of premises cited as load-bearing -- the premises without which the verdict could not be reached. If trace A uses premises 1, 3, 5 and trace B uses premises 1, 4, 5, they are NOT equivalent even if both conclude True.
  * The SAME inference rules applied at each load-bearing step (modus ponens, modus tollens, universal instantiation, existential introduction, contrapositive, etc.). A trace deriving via universal instantiation and one deriving the same fact via existential elimination are NOT equivalent.
  * The SAME key step -- the inference that actually pins down the verdict. Traces that happen to reach the same verdict via different key steps (e.g. one uses a proof-by-cases, another uses direct derivation) are NOT equivalent.
  * For "Unknown" verdicts: the SAME reason the conclusion cannot be determined. "Some doesn't entail specific individual" is NOT equivalent to "missing premise linking A to B" -- these are different logical gaps.
- Trivial relabelings ARE equivalent -- "premise 3" and "the third premise" refer to the same content. "Modus ponens" and "MP" are the same rule.
- Stylistic differences (preamble, bullet formatting, summary sentences, restatement of premises in logical notation) are IRRELEVANT. Match on the derivation content.
- A trace truncated mid-derivation must be compared only on the derivation actually performed. A truncated trace that reaches the same verdict as a complete trace by accident is NOT equivalent.

Default toward NOT_EQUIVALENT when uncertain -- a false merge is far worse than a false split for semantic-entropy clustering.

After committing to a verdict, estimate your confidence in that verdict on a [0, 1] scale. Calibrate against these anchors:

- **0.95-1.00** -- unambiguous. The traces are either word-for-word parallel on the key derivation step, or they disagree on the final answer / cite clearly disjoint evidence. No reasonable reader would flip your verdict.
- **0.80-0.94** -- confident. Small stylistic or ordering differences remain, but the load-bearing structure is either clearly shared or clearly different. A careful reader would agree with your verdict.
- **0.60-0.79** -- leaning. The traces share *some* reasoning but also differ in ways that could plausibly be considered material. You are committing to a verdict but a second reviewer might disagree.
- **0.40-0.59** -- borderline. The comparison is genuinely ambiguous: key evidence overlaps partially, or one trace is truncated, or the derivations are similar enough that reasonable readers would split. You are forced to pick a verdict but do not trust it.
- **< 0.40** -- near-random. You cannot tell whether these traces reach the same conclusion via the same derivation. Emit NOT_EQUIVALENT by default (the strictness rule above) but report your low confidence honestly.

Your confidence number must reflect the probability that your verdict would match a careful, full-information human reviewer's verdict. Do not inflate confidence to sound decisive.

End your response with exactly these two tags on their own lines:

<verdict>[EQUIVALENT or NOT_EQUIVALENT]</verdict>
<confidence>[0.00 to 1.00]</confidence>
```

F.21 LATER LLM EQUIVALENCE PROMPT: ZEBRALOGIC

Response limit: 4,096 tokens.

```
Compare two judge reasoning traces for a logic-grid puzzle (zebra-puzzle / constraint satisfaction task). The puzzle has a unique valid assignment, and the judge has picked one letter.

<reasoning_a>{reasoning_a}</reasoning_a>

<reasoning_b>{reasoning_b}</reasoning_b>

Determine whether these two traces reach the same answer via the same constraint-propagation path -- NOT merely whether they pick the same letter.

STRICT equivalence rules:
- DIFFERENT final answer letter -> NOT_EQUIVALENT, no matter how similar the reasoning is.
- Same letter but different constraint derivation -> NOT_EQUIVALENT. Specifically, the traces must agree on:
  * The SAME subset of clues used as load-bearing (the clues without which the answer could not be pinned down).
  * The SAME order of constraint propagation, up to reorderings that are provably equivalent. If one trace eliminates option X by clue 3 then clue 7, and the other eliminates X by clue 9 alone, those are NOT equivalent derivations.
  * The SAME case split structure. If trace A performs a 2-way case split on "house 2 is red or blue" and trace B does not split at all, they are NOT equivalent -- one of them is implicitly appealing to a clue the other isn't.
```

```
  * The SAME treatment of each contradicting branch. If both traces must eliminate a branch,
  ↪  they must eliminate it using the same constraint.
- Stylistic similarities (preamble, bullet formatting, summary sentences) are IRRELEVANT. Two
↪  traces with identical opening paragraphs but different clue citations are NOT equivalent.
- Trivial relabelings (e.g. "clue 3" vs "the third clue") ARE equivalent -- match on content,
↪  not citation format.
- If a trace is truncated mid-derivation, compare only the derivation that was actually
↪  performed. A truncated trace that reaches the same letter by accident is NOT equivalent to
↪  a complete trace that derives that letter.

Default toward NOT_EQUIVALENT when uncertain -- a false merge is far worse than a false split
↪  for semantic-entropy clustering.

After committing to a verdict, estimate your confidence in that verdict on a [0, 1] scale.
↪  Calibrate against these anchors:

- **0.95-1.00** -- unambiguous. The traces are either word-for-word parallel on the key
↪  derivation step, or they disagree on the final answer / cite clearly disjoint evidence. No
↪  reasonable reader would flip your verdict.
- **0.80-0.94** -- confident. Small stylistic or ordering differences remain, but the
↪  load-bearing structure is either clearly shared or clearly different. A careful reader
↪  would agree with your verdict.
- **0.60-0.79** -- leaning. The traces share *some* reasoning but also differ in ways that
↪  could plausibly be considered material. You are committing to a verdict but a second
↪  reviewer might disagree.
- **0.40-0.59** -- borderline. The comparison is genuinely ambiguous: key evidence overlaps
↪  partially, or one trace is truncated, or the derivations are similar enough that
↪  reasonable readers would split. You are forced to pick a verdict but do not trust it.
- **< 0.40** -- near-random. You cannot tell whether these traces reach the same conclusion
↪  via the same derivation. Emit NOT_EQUIVALENT by default (the strictness rule above) but
↪  report your low confidence honestly.

Your confidence number must reflect the probability that your verdict would match a careful,
↪  full-information human reviewer's verdict. Do not inflate confidence to sound decisive.

End your response with exactly these two tags on their own lines:

<verdict>[EQUIVALENT or NOT_EQUIVALENT]</verdict>
<confidence>[0.00 to 1.00]</confidence>
```