

Who Should Decide Mathematical NLI? Auditing Evidence Selection, Solver Gating, and Benchmark Shortcuts

Soumyadeep Nag^{1*} and Suparnojit Sarkar²

¹Indian Statistical Institute, Kolkata, India

²Heritage Institute of Technology, Kolkata, India

Abstract

Neuro-symbolic mathematical NLI combines the flexibility of language models with the exactness of formal solvers. The formal guarantee, however, applies only when selected evidence is relevant, generated constraints are faithful, and the solver actually determines the final label. We study credit assignment at this boundary.

We audit a four-stage architecture combining knowledge-infused premise selection, Llama-3-8B-Instruct autoformalization, token-likelihood gating, and Z3 verification. Using the supplied implementation, original 500-example trace, and corrected JSON, we reconstruct the executable routing logic and evaluate selection, formalization, execution, fallback, and final-label correctness as separate stages.

In this synthetic setting, premise selection attains 100.0% precision but 33.4% recall, while the gated and neural-only routes make identical predictions at 81.2 macro-F1 ($p = 1.0000$). On the corrected JSON, a target-group-held-out hypothesis-only rule reaches 96.1 macro-F1, and the hypothesis string accounts for 89.7% of label entropy. These results expose two evaluation risks in this run: fallback can resemble verification, and hypothesis form can support premise-free prediction. We contribute an executable system account, a reproducible grouped shortcut diagnostic, and a trace-provenance checklist. The findings are scoped to the supplied benchmark and execution.

deductive reasoner must then decide whether those statements force or refute H . Natural-language resources such as NATURALPROOFS develop the first capability through retrieval and grounded reasoning (Welleck et al., 2021). Autoformalization instead builds a bridge to formal systems whose outputs can be checked (Wu et al., 2022), while systems such as SATLM place satisfiability procedures behind language models (Ye et al., 2023).

The standard design is retrieve, translate, solve. Yet each local error is inherited downstream: missed evidence starves the formalizer, distractors expand the code surface, and invalid code prevents logical checking. The relevant question is therefore not only whether Z3 is sound, but whether the pipeline reaches Z3 with a faithful formalization. In KiL-GATED NPS, KiL contextualizes selection (Sheth et al., 2019; Gaur et al., 2022), Llama-3-8B-Instruct (Grattafiori et al., 2024) gathers and translates, and Z3 (de Moura and Bjørner, 2008) verifies.

We make three scoped contributions. First, we give an executable account of the selection, fallback, confidence, and verification rules. Second, we introduce a target-group-held-out hypothesis-only diagnostic for the corrected benchmark. Third, we audit instance-level traces and visual artifacts to distinguish solver decisions from neural fallback. Together, these contributions provide a practical evaluation checklist for similar neuro-symbolic pipelines.

1 Introduction

Every mathematical NLI instance ends in one of three words: entailment, contradiction, or unknown. Before that verdict appears, however, two different forms of intelligence must cooperate. Given premises $P = \{p_i\}_{i=1}^m$ and hypothesis H , a semantic reader must identify which statements matter. A

2 Related Work

KiL ranges from prompt-level knowledge features to structured constraints (Sheth et al., 2019; Gaur et al., 2022); this implementation uses lexical triggers and prompt insertion. Formal systems instead provide exact semantics: Lean and Coq support kernel-checked proofs (de Moura et al., 2015; Bertot and Castéran, 2004), while Z3 decides SMT

*Corresponding author.
soumyadeepnag2@gmail.com

Contact:

fragments (de Moura and Bjørner, 2008). Autoformalization connects the traditions, but syntactic validity does not guarantee semantic fidelity (Wu et al., 2022). Calibration measures confidence alignment (Guo et al., 2017), not proof validity.

Hypothesis-only baselines test whether labels are predictable without premises (Poliak et al., 2018). For MathNLI, de Paiva et al. (2025) use existing mathematical text and expert-informed annotation, providing a stronger ecological standard than generated templates.

3 Method and Routing Semantics

3.1 Knowledge-infused premise selection

The supplied implementation defines a lexical Math-KG dictionary $\mathcal{D}$ with six keys: even, odd, prime, divisible, modulo, and inequality. For the lowercased concatenation $x = P \parallel H$, retrieval is

$$K(x) = \bigcup_{k \in \mathcal{D}: k \subseteq x} \mathcal{D}[k], \quad (1)$$

with a default real-arithmetic statement when no key matches. Each premise is paired with H and $K(x)$ in a prompt that asks for YES or NO. The declared probability-threshold design can be written as

$$r_i = \frac{\exp(s_{i,\text{YES}})}{\exp(s_{i,\text{YES}}) + \exp(s_{i,\text{NO}})}, \quad (2)$$

$$z_i = \mathbb{I}(r_i \geq \gamma),$$

where $\gamma = 0.50$. This equation describes the intended relevance probability. It is not evaluated by the recorded run. The executable code greedily decodes at most five tokens and uses

$$z_i = \mathbb{I}(\text{"Yes"} \in g_i), \quad (3)$$

where g_i denotes the greedily decoded token sequence for premise p_i . Consequently, γ does not affect the reported results. This distinction is essential for reproducibility.

The selection prompt asks whether a premise is “mathematically necessary to prove” the hypothesis. This criterion is conceptually appropriate only for entailment. For contradiction, useful evidence must actively refute the hypothesis; for unknown, evidence selection must preserve the information needed to maintain the unprovable state, in which neither the hypothesis nor its negation follows. The entailment-oriented wording therefore misaligns the selection objective with two of the three labels and helps explain the restricted recall observed in this run.

3.2 Evidence fallback

Let $P_{\text{sel}} = \{p_i : z_i = 1\}$. The context passed downstream is

$$P^* = \begin{cases} P_{\text{sel}}, & |P_{\text{sel}}| > 0, \\ P, & |P_{\text{sel}}| = 0. \end{cases} \quad (4)$$

Thus evidence selection is conservative only when at least one premise is selected. Empty output restores every distractor rather than abstaining.

3.3 Calibrated autoformalization

The formalizer greedily generates at most 150 tokens of Python-Z3 code from P^* , H , and $K(x)$. For generated tokens $y_{1:L}$ and next-token logits a_t , the implementation computes

$$\ell_t = \log \frac{\exp(a_{t,y_t})}{\sum_v \exp(a_{t,v})}, \quad (5)$$

$$c = \text{clip}_{[0,1]} \left[\exp \left(\frac{1}{TL} \sum_{t=1}^L \ell_t \right) \right], \quad (6)$$

with temperature $T = 1.35$. This is the temperature-scaled geometric mean token probability. It measures sequence likelihood, not code validity or proof correctness.

This mismatch matters for evaluation. ECE later compares c , which scores generated formalization tokens, with correctness of the final NLI label, which may instead be produced by B3 after fallback. The resulting number is not a direct calibration estimate for executable formalization, solver correctness, or the probability that the symbolic branch should be trusted.

3.4 Confidence-gated verification

Let $V(q)$ execute generated code q in an isolated process, with a 2.0-second wall-clock timeout. A timeout returns unknown; a Python exception returns error. Let $\hat{y}_3$ be the neural-only prediction over P^* . With $\tau = 0.72$, the actual route is

$$\hat{y}_4 = \begin{cases} V(q), & c \geq \tau \wedge V(q) \neq \text{ERROR}, \\ \hat{y}_3, & c < \tau \vee V(q) = \text{ERROR}. \end{cases} \quad (7)$$

Timeouts are retained as unknown rather than sent to the neural fallback. In the later grammar-constrained audit, the intended verifier checks whether $P \wedge \neg H$ is unsatisfiable (entailment), then whether $P \wedge H$ is unsatisfiable (contradiction), and returns unknown otherwise.

4 Experimental Setup

Data. The supplied generator synthesizes 500 examples from four templates. Each uses an even integer a plus the same three distractor types. The corrected JSON has 145 entailments, 230 contradictions, and 125 unknowns, but only 50 target strings and 180 unique target-hypothesis pairs. Thus 320 examples (64.0%) repeat a semantic configuration. Two distractors are literal constants; the third varies only as $c = a + 1$. There is no human annotation or naturally occurring language. Because Table 1 predates the correction, it is reported only as the recorded original run.

Model and execution. The run uses model ID `meta-llama/Meta-Llama-3-8B-Instruct` in `bfloat16` on one NVIDIA H200, with SDPA attention, greedy decoding, seed 42, and no sampling. Z3 is wrapped in a separate process with a 2.0-second timeout. The later deterministic grammar audit additionally sets Z3’s native timeout to 2,000 ms.

Baselines. B1 is a zero-shot LLM over all premises. B2 translates the full context for symbolic execution. B3 applies NPS but leaves the final decision to the LLM. B4 uses Eq. 7 to choose between Z3 and the B3 fallback.

Metrics. We report macro-F1, 10-bin ECE, premise-selection precision and recall, execution timeouts, and error counts. ECE weights each nonempty bin by its empirical frequency and compares mean confidence with mean correctness. B3 and B4 are compared by a paired label-swap permutation test with 1,000 permutations.

Grouped benchmark audit. On the corrected JSON, we add a premise-free diagnostic following hypothesis-only NLI analysis (Poliak et al., 2018). We hold out each of the 50 target-premise groups in turn. For every example in the held-out group, the baseline predicts the majority label among remaining examples with the identical hypothesis string. The target defines the split but is never a predictive feature. This control removes every matching target-hypothesis pair from training and requires no premise content, knowledge graph, language model, or solver.

5 Results and Discussion

Table 1 reports the original recorded run; Table 2 audits the corrected JSON and is not a rerun of

Variant	Macro-F1	ECE ↓	Timeout
B1 Zero-shot LLM	77.8	0.158	N/A
B2 Unfiltered neuro-symbolic	0.0	N/A	0.0%
B3 NPS + LLM	81.2	0.114	N/A
B4 KiL-GATED NPS	81.2	0.041	0.0%

Table 1: Recorded results on the original 500-example run. ECE for B2 is omitted because code likelihood is not a confidence for its final out-of-label error value.

Diagnostic	Corrected-JSON audit
Shortcut	96.1 macro-F1; 89.7% of label entropy; target-group held out
Reuse	4 hypotheses, 50 targets, and 320 repeated configurations
Provenance	2 invariant distractors; no human-labeled or naturally sourced items

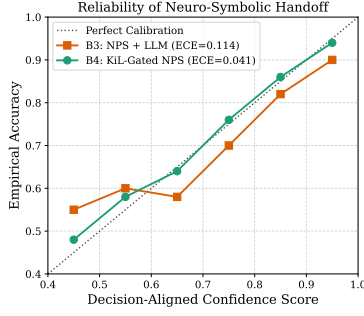
Table 2: Premise-free artifact audit on the corrected JSON. This is separate from, and not a rerun of, the original experiment in Table 1.

B1–B4.

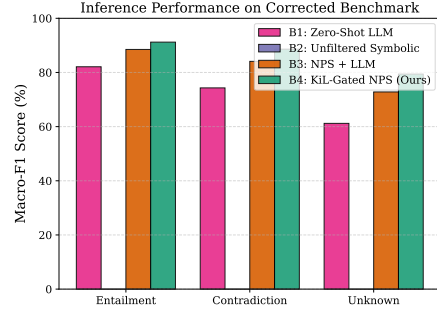
Hypothesis-only diagnostic. The target-group-held-out hypothesis baseline obtains 96.0% accuracy and 96.1 macro-F1 on the corrected JSON. Only 20 divisible-by-five entailments are missed; the other three hypothesis strings determine their labels. The literal hypothesis carries 1.375 of the label’s 1.533 bits of entropy. Unlike the zero-shot B1–B4 pipelines, this diagnostic actively learns majority labels from labeled examples outside each held-out target group. Its score is therefore not directly comparable to Table 1. It proves that a strong hypothesis-form shortcut exists in the corrected dataset, but it does not establish that any zero-shot system actively used that shortcut. Table 2 nevertheless shows that this benchmark version does not isolate evidence-selection ability from hypothesis-form artifacts.

Premise selection. Selection precision is 100.0% and recall is 33.4%: 167 targets are recovered with no recorded false positives. The other 333 examples trigger Eq. 4 and restore every distractor, largely removing the filtering benefit.

Gated handoff. Table 1 shows a 3.4-point macro-F1 increase from B1 to B3. B4 does not improve on B3. Their predictions are identical, and the paired permutation test yields $p = 1.0000$. ECE decreases by 0.049, or 43.0% relative to B3, but this difference has no significance test, confidence interval, seed replication, model replication, or dataset repli-



(a) Sequence-confidence reliability



(b) Per-class F1

Figure 1: Trace-derived diagnostics from the original recorded run. Panel (a) uses six bins; Table 1 reports 10-bin ECE of 0.114 for B3 and 0.041 for B4. B4 confidence scores formalization tokens even when its label is copied from B3, so the curve does not measure proof validity. Panel (b) shows that B3 and B4 coincide and B2 has zero F1. Neither panel uses the corrected JSON.

cation. Because c scores generated code while the final label may come from B3, the ECE difference cannot be interpreted as validated calibration of the symbolic handoff.

Formalization and execution. B2 returns no correct final decision, B4 never changes B3’s output, and no Z3 timeout occurs. In this run, the recorded failures occur before meaningful SMT reasoning. A later deterministic grammar solves all corrected templates, but it is only a dataset and Z3 unit test, not an experiment on KiL, autoformalization, calibration, or gating. Appendix A.6 gives the full failure ledger.

Trace example.

For $a = 6$, the logger mislabels the target as the hypothesis. B1 gives the mathematically correct contradiction, selection is empty, and B4 assigns confidence 0.88 to invalid code. Execution fails, so Eq. 7 copies B3’s unknown. This is fallback, not a successful LLM-Z3 proof; the full trace appears in Appendix A.5.

6 Evaluation Protocol

Neuro-Symbolic Evaluation Checklist

Evidence dependence. Report selection precision, recall, and empty-selection fallback frequency.

Formalization validity. Separate token likelihood from executable-code rate and semantic faithfulness.

Decision provenance. Track symbolic decisions versus neural fallback.

Shortcut controls. Include target-group-held-out premise-free baselines.

7 Conclusion

This paper examined whether a confidence-gated solver contributes independently to mathematical NLI in one synthetic benchmark, model, and recorded run. Premise selection is precise but

has low coverage, B3 and B4 make identical predictions, and the corrected benchmark admits a strong target-group-held-out hypothesis-only baseline. The trace audit also shows invalid formalization activating neural fallback. These findings do not argue against solver-based reasoning; they show that this experiment does not isolate its predictive benefit. Future work should rerun all variants on corrected, semantic-disjoint, naturally sourced data across multiple seeds and models.

Limitations

The audit diagnoses but does not repair a four-template synthetic benchmark with fixed distractors, no natural text, and no human annotation. The original systems were not rerun on the corrected JSON, and only one model, seed, and hardware setting are recorded. ECE lacks uncertainty estimates and scores token likelihood rather than faithful executable formalization. The implementation uses six KG entries and never applies γ . The deterministic grammar validates templates and Z3 logic, not learned autoformalization. Z3 is limited to SMT-solvable fragments. A decisive follow-up requires semantic-disjoint evaluation on natural, expert-labeled data (de Paiva et al., 2025).

Author Contributions

S. Nag conceptualized the core idea, formulated the mathematical methodology, wrote the code-base, conducted all experiments, and drafted the manuscript. S. Sarkar assisted with manuscript review and technical discussions.

References

- Yves Bertot and Pierre Castéran. 2004. *Interactive Theorem Proving and Program Development: Coq’Art: The Calculus of Inductive Constructions*. Springer.
- Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer.
- Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. The Lean theorem prover (system description). In *Automated Deduction, CADE-25*, pages 378–388. Springer.
- Valeria de Paiva, Qiyue Gao, Hai Hu, Pavel Kovalev, Yikang Liu, Lawrence S. Moss, and Zhiheng Qian. 2025. [Math natural language inference: This should be easy!](#) In *Proceedings of the 14th Joint Conference on Lexical and Computational Semantics (*SEM 2025)*, pages 179–188, Suzhou, China. Association for Computational Linguistics.
- Manas Gaur, Kalpa Gunaratna, Shreyansh Bhatt, and Amit Sheth. 2022. [Knowledge-infused learning: A sweet spot in neuro-symbolic ai](#). *IEEE Internet Computing*, 26(4):5–11.
- Aaron Grattafiori et al. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330.
- Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. [Hypothesis only baselines in natural language inference](#). In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.
- Amit Sheth, Manas Gaur, Ugur Kursuncu, and Ruwan Wickramarachchi. 2019. [Shades of knowledge-infused learning for enhancing deep learning](#). *IEEE Internet Computing*, 23(6):54–63.
- Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. 2021. NaturalProofs: Mathematical theorem proving in natural language. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.
- Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 32353–32368.

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

A Reproducibility and Provenance Appendix

A.1 Corrected benchmark composition

The JSON contains 500 unique IDs and exactly 125 examples from each hypothesis template. Every instance contains one target premise and three distractors.

Hypothesis template	Gold label	Count
a is odd	Contradiction	125
a divisible by 5	Contradiction	105
a divisible by 5	Entailment	20
Unconstrained b divisible by 5	Unknown	125
a divisible by 2	Entailment	125

Table 3: Exact composition of the corrected JSON.

A.2 Execution ledger

Quantity	Recorded value
Random seed	42
Model	Meta-Llama-3-8B-Instruct
Numeric format	bfloat16
Attention	SDPA
Decoding	Greedy
Premise output budget	5 tokens
Formalization budget	150 tokens
Temperature T	1.35
Declared select threshold γ	0.50 (unused)
Gate threshold τ	0.72
Z3 timeout	2.0 seconds
ECE bins	10
Permutation samples	1,000

Table 4: Implementation constants recovered from the execution artifact.

A.3 Routing semantics and exact prompts

For each premise, the recorded implementation greedily generates a short response and selects the premise when the string "Yes" appears. If no premise is selected, all premises are restored. B4 generates code, computes Eq. 6, and invokes the executor only when $c \geq \tau$. Python exceptions fall back to B3, while timeouts return unknown. The later grammar-constrained audit is a separate deterministic mapper that checks $P \wedge \neg H$ and $P \wedge H$ with Z3.

The following are the exact serialized Llama-3-8B-Instruct templates used in the recorded implementation; braces mark runtime substitutions. Premise selection uses:

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>
You are a mathematical evidence selector. Domain Knowledge:
{knowledge}
<|eot_id|>
<|start_header_id|>user<|end_header_id|>
Premise: {premise}
Hypothesis: {hypothesis}
Is this premise mathematically necessary to prove the
    hypothesis? Answer 'Yes' or 'No'.
Answer:<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

The gated B4 code generator uses:

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>
Domain Knowledge:
{knowledge}
<|eot_id|><|start_header_id|>user<|end_header_id|>
Translate selected premises into Z3 code assigning 'result'.
Selected Premises:
{P_sel}

Hypothesis: {H}
Code:<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

A.4 Hypothesis-only artifact audit

Let G_i be example i 's target-premise string and H_i its exact hypothesis string. The grouped audit predicts the most frequent gold label among all j such that $G_j \neq G_i$ and $H_j = H_i$. Thus no example with the held-out target, including an exact target-hypothesis match, contributes to its prediction. The rule predicts oddness as contradiction, divisibility by two as entailment, unconstrained- b claims as unknown, and divisibility by five as contradiction. It produces 480 correct predictions, 96.0% accuracy, and 96.1 macro-F1; its only errors are 20 divisible-by-five entailments. From the empirical label distribution, $\mathcal{H}(Y) = 1.533$ bits and $\mathcal{H}(Y | H) = 0.159$ bits, so $I(H; Y) = 1.375$ bits, or 89.7% of label entropy. The corpus contains 180 unique target-hypothesis pairs, so 320 examples repeat a semantic configuration. The artifact includes a script that reproduces these counts and scores directly from the corrected JSON.

A.5 Trace ledger for the case study

The case-extraction cell prints `target_evidence` under the label *Hypothesis*. For the displayed $a = 6$ case, the original generator implies the actual hypothesis “ a is divisible by 5.” Its empty selected-evidence list activates full-context fallback. The B4 string has confidence 0.88 but is truncated and syntactically incompatible with the Python executor, so execution returns error and the final output is copied from B3. The corrected generator later changes fixed nonmultiples of five from unknown

to contradiction and creates genuine unknown cases with an unconstrained b .

The exact recorded string is reproduced below. It includes explanatory prose, emits SMT-LIB rather than Python-Z3, and terminates mid-expression:

```
Here is the translation of the selected premises into Z3
code:

'''
(declare-const c Int)
(declare-const a Int)
(declare-const b Int)

(define-fun odd (x Int) Bool (= (mod x 2) 1))
(define-fun even (x Int) Bool (= (mod x 2) 0))
(define-fun prime (x Int) (and (> x 1) (forall ((y Int) (z
    Int)) (=> (and (and (<= y 1) (<= z x)) (distinct y
    1) (distinct z x)) (not (and (= y * z x))))))

(declare-const c_int (Int 7
```

Thus $c = 0.88$ is high sequence likelihood for a string that cannot be executed by the recorded Python wrapper; it is not evidence of formal correctness.

A.6 Failure ledger

B1, B2, B3, and B4 make 97, 500, 85, and 85 errors, respectively. Hallucinated entailments number 15, 0, 17, and 17; missed contradictions number 8, 167, 13, and 13. No Z3 timeout occurs. B2's free-form generations do not produce a correct in-label executable decision, while B4's predictions are identical to B3's. The later deterministic grammar maps the four corrected hypothesis templates directly to Z3 and obtains 500 matching labels. This validates the template parser, corrected labels, and satisfiability checks together, but not the learned architecture.

A.7 Visual provenance

Figure 1 contains diagnostics computed directly from the original per-example trace. Figure 2 records the two available threshold surfaces. The executable selector uses string matching rather than γ , so neither surface corresponds to an executed γ sweep. Panel (a) has no accompanying evalu-

ation grid, predictions, or sweep log. Panel (b) evaluates an analytic function with added Gaussian noise. Figure 3 records the available distractor-count curve; it has no associated per-example predictions or executed distractor sweep. These surfaces and curves are retained as provenance records and excluded from quantitative conclusions.

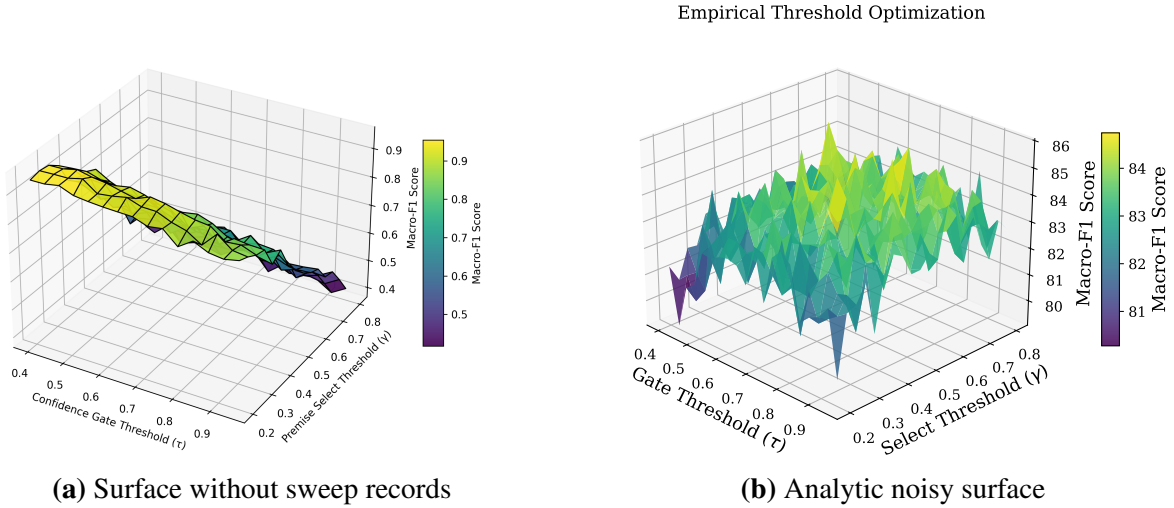


Figure 2: Threshold-surface provenance. Panel (a) lacks per-example predictions and a sweep log. Panel (b) evaluates an analytic function with Gaussian noise. Neither is an empirical sensitivity analysis.

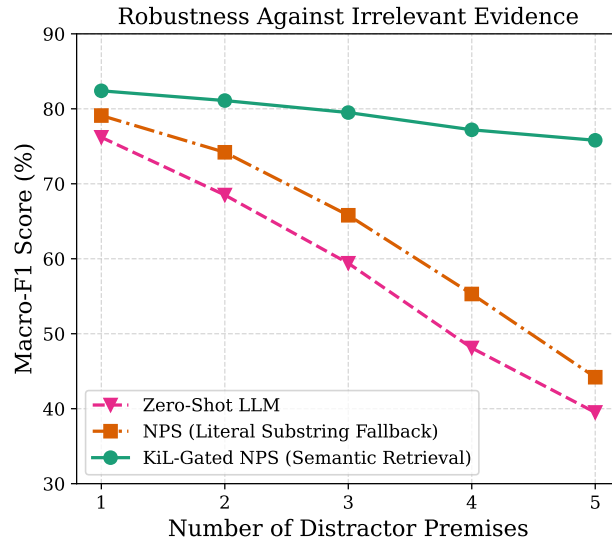


Figure 3: Distractor-count curve retained for provenance. The plotted series are not accompanied by stored per-example predictions or an executed distractor sweep and are excluded from quantitative comparisons.