MathNLP 2026

MathNLP 2026: The 4th Workshop on Mathematical Natural Language Processing

Proceedings of the Workshop

October 29, 2026

©2026 Association for Computational Linguistics

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
317 Sidney Baker St. S
Suite 400 - 134
Kerrville, TX 78028
USA
Tel: +1-855-225-1962
acl@aclweb.org

ISBN 979-8-89176-468-2

MathNLP 2026

The articulation of mathematical arguments is a fundamental part of scientific reasoning and communication. Across many disciplines, expressing relations and interdependencies between quantities is at the centre of scientific argumentation. Nevertheless, despite its importance, the application of contemporary NLP models for inference over mathematical text remains under-explored or subject to important limitations. MathNLP represents a forum for discussing new ideas to advance research on Mathematical Natural Language Processing, welcoming novel contributions on model architectures, evaluation methods and downstream applications.

Recent advances in Natural Language Processing (NLP) enabled by Deep Learning-based architectures bring the opportunity to support the interpretation of textual content at scale. The application of these methods can facilitate scientific discovery, reducing the gap between current research and the available large-scale scientific knowledge. Previous work has shown the potential of designing neural architectures for different mathematical natural language inference tasks, such as premise selection in natural language, expression derivation, and mathematical information retrieval.

However, there are still technical gaps that need to be addressed such as the availability of datasets and evaluation tasks, techniques for the joint interpretation of different modalities present in mathematical text (equational and natural language), the understanding of unique aspects of mathematical discourse and multi-hop models for mathematical inference.

For additional details about MathNLP 2026, please visit the website: https://sites.google.com/view/mathnlp2026.

Organizing Committee

General Chair

Marco Valentino, University of Sheffield

Program Chairs

Deborah Ferreira, The MathWorks
Mokanarangan Thayaparan, The MathWorks
Leonardo Ranaldi, University of Edinburgh
Jo-Ku Cheng, University of Sheffield
Andre Freitas, Idiap Research Institute and University of Manchester

Program Committee

Program Committee

Joshua Ong Jun Leang
Feng He
Zhonglin Xie
Mingzi Cao
Yanyu Chen
Yinjie Cheng
Yanwen Peng
Jiajun Chen
Jingyun Wang
Jingyang Deng
Hikaru Matsuoka
Constantinos Karouzos
Koji Mineshima
Jordan Meadows
George Chrysostomou
Carlos Toxtli
Xin Quan
Xingwei Tan
Lan Zhang
Jasivan Alex Sivakumar
Lei Xu
Somak Aditya
Surangika Ranathunga
Dhairya Dalal
Qianying Liu
Fajri Koto
Nischal Reddy Chandra
Pierre Beckmann
Cheuk Lun Ko

Table of Contents

Beyond Better Helpers: A Shared-Kernel Study of Symbolic Interfaces for Auditable LLM Algebra
Norio Kosaka, Genki Osada, Shinichi Higashino and Shuji Yamaguchi 1

Hard or Just Unreached? Diagnosing the Sampling Blind Spot in Math-Reasoning Difficulty Estimation
Luca Zhou, Sajel Shah, Emanuele Rodolà and Roberto Dessi 13

BlueprintRepair: Typed Local Edits for Failed Lean Proof Blueprints
Ruslan Khrulev 30

From Recognition to Reconstruction: Towards Robust Procedural Mathematical Reasoning
Fateme Mazdarani and Carlos Toxtli 42

Elision Markers in Mathematical Prose: Language Models Respond to Claimed Effort Rather Than Hidden Derivation Depth
Md. Faisal Sheikh 57

AMIR-GRPO: Inducing Implicit Preference Signals into GRPO
Amir Hossein Yari and Fajri Koto 69

The Cost of Incoherent Context Grows with Capability: A Control for Exemplar Retrieval
Srinivas Harish 86

InvariantSpec: Benchmarking LLM Autoformalization of Conservation and Dissipation Laws into Verifiable State-Space Constraints
Hikaru Matsuoka 101

MathQueue: Reasoning Scheduling Helps Only Within a Narrow Budget, and Answer Extraction Decides Whether It Helps at All
Jonghyun Lee and Byungjun Kim 112

Typed Neuro-Symbolic Autoformalization of Quantitative Clinical Language: Executable Constraints and Selective Verification
Aleksandra Beliaeva 126

Which Mathematical Context Supports Target-Aligned Conjecture Generation? A Paired Ablation Study
Austin LaHue, Michael Burr, Luis David Garcia Puente, Vinita Gangaram Jansari, Benjamin E Nye and Carlos Toxtli Hernandez 136

The Connective but in Informal Mathematical Proof: A Corpus-Based Study
Rinta Yamada, Sakiko Yamasaki, Risako Ando, Takayuki Amamoto, Ryota Akiyoshi and Koji Mineshima 152

NaturalPRISM: A Natural Language Premise Retrieval Task for Research-level Intermediate Proof States
Harris Proctor, LI AN, Austin LaHue, Michael Burr, Vinita Gangaram Jansari, Carlos Toxtli Hernandez, Luis David Garcia Puente and Benjamin E Nye 167

Improving Natural-Language Combinatorial-Optimization Accuracy in Resource-Constrained Language Models via Formal Abstractions
Shrenil Shaun Sharma and Avi Sharma 177

Structural Signatures of Correct Mathematical Reasoning Traces
Maria Marina and Alexander Panchenko 194

Simulating Student-Tutor Interactions with LLMs for Multilingual Math Feedback
Junior Cedric Tonga, Kv Aditya Srivatsa, Kaushal Kumar Maurya, Fajri Koto and Ekaterina Kochmar 201

Beyond Accuracy: Diagnosing Algebraic Reasoning Failures in LLMs Across Nine Complexity Dimensions
Parth Shrikant Patil, Yash Sinha, Murari Mandal and Dhruv Kumar 216

Same Text, Different Transitions: Action Insufficiency in Lean Provers
Edison Yang, Neel Marripalapu, Akshaj Agadi and Andrew Kim 240

Scoring-Location Sensitivity in Process Reward Models
Jai Kumar Sharma 254

Embedding-Based Educational Taxonomies
Kia Karbasi, Kaveh Karbasi and Gregory Pottie 268

Numbers Through Signs: How LLMs Connect Counts, Number Words, and Numerals
J. Albert Yoo 285

Definition Modeling for Mathematics: Understanding Audience Bias and its Problem Solving Relevance
Carolina Dias-Alexiou, Edison Marrese-Taylor, A. Solano and Yutaka Matsuo 291

Who Should Decide Mathematical NLI? Auditing Evidence Selection, Solver Gating, and Benchmark Shortcuts
Soumyadeep Nag and Suparnojit Sarkar 305

Proof Identity and Divisible Correspondence in Multi-Derivation Reasoning Evaluation
Toprak Seda Karaosmanoğlu 312

Beyond Better Helpers: A Shared-Kernel Study of Symbolic Interfaces for Auditable LLM Algebra

Norio Kosaka **Genki Osada** **Shinichi Higashino** **Shuji Yamaguchi**
LY Corporation
Tokyo, Japan

Abstract

Large language models can delegate algebra to external tools, but improvements attributed to a structured tool interface may instead come from the symbolic helper code hidden behind that interface. We isolate these factors on polynomial systems by holding the model, tasks, backend, and symbolic kernel fixed. Using Qwen3-8B, we compare text-only reasoning, unrestricted Python/SymPy generation, generated Python calling the same helper functions as the structured system, and a typed diagnosis–solve–check interface. On 36 tasks, the typed interface obtains 25/36 correct answers, compared with 14/36 for matched-helper code, 17/36 for raw SymPy, and 16/36 for text only. It also produces 23/36 automatically verified-correct answers versus 4/36 for the matched-helper control, and contains no execution-error runs versus 29/36. The largest separation occurs on exact finite-set tasks, where the typed interface solves 8/12 systems and both code conditions solve 0/12. Access to packaged helper functionality alone therefore does not reproduce the typed condition's performance. The typed interface also removes the need to generate imports, API calls, control flow, and result-handling code, so the comparison measures end-to-end interface usability rather than mathematical reasoning or schema syntax in isolation. We interpret the result as a deployed-interface effect for one model and a development-informed task suite.

1 Introduction

Mathematical language agents increasingly move between natural-language problem statements, formal equations, executable programs, symbolic tools, and machine-checkable conclusions (Yan et al., 2025; Gao et al., 2023; Gou et al., 2024; Pan et al., 2023; Liu et al., 2025). External computation can improve answer accuracy, but execution alone does not make a result auditable. A solver-backed response may still omit solutions, ignore a domain restriction, hide a failed computation, or report candidates without retaining evidence that supports the final claim (Bagheri Nezhad et al., 2026; Zhang et al., 2024; Liu et al., 2025).

Polynomial systems provide a compact setting in which these obligations can be separated. Candidate assignments can be checked by substitution, structural claims can be assessed with computer algebra, and finite-set answers expose a crucial distinction: zero residuals establish that the listed points are valid, but do not establish that no solutions were omitted (Brake et al., 2016; Sommese et al., 2005; Cox et al., 1997). This makes polynomial systems useful not only for measuring final-answer accuracy, but also for evaluating whether a retained trace validly covers the claim being asserted.

A central experimental difficulty is that tool-interface comparisons often change two things at once. A typed tool may expose constrained arguments and a structured workflow, but also provide normalization, solver wrappers, domain handling, residual computation, and exception handling that a free-form-code agent must synthesize itself. A typed-versus-raw-code comparison therefore cannot determine whether gains come from the model-facing interface or from better packaged symbolic functionality. We address this confound with a *matched-helper code* condition: the model writes ordinary Python, but imports and calls the exact same versioned symbolic kernel used by the typed actions. The primary comparison thus holds helper implementation fixed while changing how the model invokes it and receives results.

We ask three questions. **RQ1** asks whether packaged symbolic helpers improve over unrestricted Python/SymPy generation. **RQ2** asks whether, with the helper implementation fixed, a typed interaction format changes correctness and automatically verified correctness relative to generated Python. **RQ3** asks how the interfaces change ex-

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 1–12
October 29, 2026 ©2026 Association for Computational Linguistics

plicit verification, execution failures, runtime, and performance across task families.

Using `qwen3:8b` through Ollama (Qwen Team, 2025; Ollama, 2026), we evaluate four conditions on the same 36 polynomial-system tasks. The typed interface obtains 25/36 correct answers, compared with 14/36 for matched-helper code, 17/36 for raw SymPy, and 16/36 for text only. It yields 23/36 automatically verified-correct answers versus 4/36 for matched-helper code, while reducing runs containing an execution error from 29/36 to 0/36. The largest gap occurs on exact finite-set tasks: the typed interface solves 8/12, whereas both code conditions solve none. Access to the shared helpers alone therefore does not reproduce the typed condition's performance. The remaining contrast includes constrained invocation, workflow cues, structured statuses, evidence serialization, and the removal of generated glue code; it does not isolate improved mathematical reasoning or any one interface component. Earlier oracle-equation and restricted-action pilots, retained as secondary analyses, further test whether language grounding or the available action surface explains this pattern.

Contributions. This paper makes three contributions:

- **A shared-helper controlled evaluation.** We introduce a matched-helper code baseline that invokes the same symbolic kernel as the typed interface, separating packaged-helper effects from the model-facing interaction format.
- **Claim-aware evaluation.** We distinguish answer correctness, candidate validity, set completeness, explicit verification, execution failure, and automatically verified-correct answers for finite sets, witnesses, and structural diagnoses.
- **Evidence for a deployed-interface effect.** Under a fixed model, task suite, backend, and helper implementation, the typed interface is associated with higher correctness and retained claim support while avoiding the execution failures that dominate generated matched-helper code. This is an end-to-end system result, not evidence that typing alone improves mathematical reasoning.

2 Related Work

Program-aided and neuro-symbolic mathematical reasoning. Program-aided methods separate linguistic decomposition from external computation. PAL delegates computation to generated Python programs (Gao et al., 2023), Program of Thoughts uses executable programs to separate reasoning from calculation (Chen et al., 2023), and ToRA interleaves language-model reasoning with external mathematical tools (Gou et al., 2024). Related neuro-symbolic systems translate natural language into formal representations executed by a solver, including Logic-LM (Pan et al., 2023) and LINC (Olausson et al., 2023). MCP-Solver similarly exposes constraint, SAT, and SMT systems through a standardized tool interface (Szeider, 2025). These works establish the value of external symbolic computation. Our focus is complementary: rather than comparing tool use with no tool use alone, we ask whether the model-facing interface matters when the symbolic helper implementation is held fixed.

Verifiable code and symbolic checking. SymCode prompts an LLM to generate self-contained SymPy programs, executes them, and uses interpreter feedback for verification and self-debugging (Bagheri Nezhad et al., 2026). Our raw-code condition follows this general reasoning-as-code paradigm, while our matched-helper condition provides a stronger control: generated Python can call the same normalization, diagnosis, solving, domain, and checking functions used by the typed interface. Work on symbolic verification of intermediate reasoning likewise demonstrates the value of machine-checkable representations for process evaluation (Zhang et al., 2024). SAFE uses Lean to verify step-level mathematical claims (Liu et al., 2025), and tool-integrated Lean validation has also been used to generate and align synthetic mathematical data (Obozov et al., 2025). Our setting is less expressive than a proof assistant, but permits exact candidate checks and explicit completeness obligations for a controlled class of algebraic claims.

Tool-using agents and structured action spaces. ReAct formalized the reasoning–action–observation loop (Yao et al., 2023), while Toolformer and Gorilla studied tool invocation and API use (Schick et al., 2023; Patil et al., 2023). Structured actions can reduce syntax and serialization failures, but comparisons against

raw code usually confound action format with the functionality packaged behind the API. Our four-condition design decomposes this comparison. Raw SymPy versus matched-helper code probes the effect of packaged symbolic functionality; matched-helper code versus typed actions probes the deployed interaction format with the helper kernel fixed. The latter still jointly changes schema constraints, workflow guidance, status presentation, and evidence serialization, so we do not interpret it as a schema-syntax-only intervention.

Positioning within MathNLP. The tasks require a model to interpret variables, equations, domains, and answer modes before producing a checkable mathematical claim. The paper is therefore most closely related to neuro-symbolic mathematical inference and evaluation of tool-supported mathematical reasoning. It does not propose a new computer-algebra algorithm or a full autoformalization system. Its contribution is a controlled evaluation of the interface layer through which mathematical language agents exchange formal problems, candidate solutions, structural statuses, and verification evidence with symbolic computation.

3 Task Setting and Symbolic Interfaces

Problem setting. A task consists of a mathematical prompt u, a variable list $V = (x_1, \ldots, x_n)$, a declared solution domain $D \in \{\mathbb{R}, \mathbb{C}\}$, and polynomial equations

$$F = \{f_i(x_1, \ldots, x_n) = 0\}_{i=1}^{m}. \qquad (1)$$

The required answer mode is one of the following:

- ALL-SOLUTIONS: return the complete finite solution set in D;
- WITNESS: return at least one assignment satisfying every equation and domain restriction; or
- DIAGNOSIS: return a domain-aware structural label.

For diagnosis tasks, algebraic inconsistency is distinguished from absence of real solutions: a system may have complex solutions but no point in $\mathbb{R}$. The present study isolates tool-mediated symbolic reasoning after the task variables, equations, domain, and answer mode have been specified; it does not evaluate open-ended autoformalization.

Verification obligations. For a returned assignment s, *candidate validity* requires $f_i(s) = 0$ for every original equation and satisfaction of the declared domain. This is sufficient for an WITNESS claim. For an ALL-SOLUTIONS answer $\widehat{S}$, candidate validity establishes only soundness of the listed points. The claim additionally requires *set completeness*: every solution in D must belong to $\widehat{S}$. Consequently, residual checks do not by themselves establish an all-solutions claim. A run is counted as verified-correct only when its final answer is correct and its retained evidence covers the obligations associated with its answer mode.

Shared symbolic kernel. The matched-helper and typed conditions invoke one versioned symbolic kernel. The kernel implements equation normalization, structural diagnosis, candidate generation, domain filtering, residual computation, and candidate checking. It separates execution, candidate-validity, domain-validity, diagnosis, and completeness statuses. In particular, successful execution or zero residuals for returned candidates are not automatically treated as evidence that a finite solution set is complete.

Model-facing interfaces. The raw-code condition writes unrestricted Python/SymPy and is responsible for constructing solver calls, checks, and output handling. The matched-helper condition also writes ordinary Python, but may import the shared kernel functions directly. The typed condition accesses the same kernel through three schema-constrained actions:

1. `diagnose_groebner`, which returns supported structural conclusions such as inconsistency or zero-dimensionality;
2. `solve_polynomial_system`, which returns normalized candidates, domain annotations, residual information, and a conservative completeness status; and
3. `check_solution`, which substitutes a proposed assignment into the original equations and reports symbolic residuals and domain validity, but never certifies completeness of an enumerated set.

The typed adapter performs request validation and structured serialization, but delegates the mathematical operations to the same kernel used by matched-helper code.

Trace semantics. Every tool-enabled run retains generated code or typed requests, helper inputs and outputs, structured statuses, warnings, errors, and the final response. Execution failures and unsupported cases remain visible rather than being converted into ordinary answers. Evidence is linked to the final claim rather than credited merely because a solver or checker was invoked.

4 Experimental Design

Research questions. We study three questions. **RQ1** asks whether providing a shared symbolic helper layer improves over unrestricted Python/SymPy generation. **RQ2** asks whether, with helper functionality fixed, typed invocation changes answer correctness and automatically verified correctness relative to generated Python. **RQ3** asks how interface design changes explicit verification, execution failures, runtime, and performance across task families.

Conditions. We evaluate four conditions on every task:

1. **Text only** (`text_only`): the model receives the task and no external computation.
2. **Raw SymPy** (`raw_sympy`): the model writes and executes unrestricted Python/SymPy.
3. **Matched-helper code** (`matched_helper_code`): the model writes ordinary Python that imports the same normalization, diagnosis, solving, domain, and checking functions used by the typed condition.
4. **Typed helper interface** (`typed_helper_interface`): the model invokes schema-constrained diagnosis, solve, and check actions backed by the shared kernel and receives structured status and evidence records.

All conditions receive the same equations, variables, domain, answer-mode specification, and final-answer requirement. Conditions 2–4 use SymPy as the underlying algebra system. Conditions 3 and 4 additionally use the exact same versioned symbolic-kernel implementation. Their intended difference is the model-facing interaction format: generated Python and free-form result handling versus validated action calls and structured responses. Workflow wording, status presentation, and evidence serialization necessarily differ with this format and are therefore part of the deployed-interface treatment. The study does not include an additional code condition supplied with a working import and invocation template; consequently, it does not separate low-level API scaffolding from the broader structured interaction protocol.

Primary contrasts. Raw SymPy versus matched-helper code probes whether packaged symbolic functionality is sufficient to improve generated-code reasoning. Matched-helper code versus the typed interface is the primary controlled contrast: helper implementation is fixed, while invocation, workflow affordances, and result serialization change. Text only provides context for the value of external computation rather than a matched causal control.

Model and execution environment. All runs use `qwen3:8b` through Ollama (Qwen Team, 2025; Ollama, 2026), with temperature 0, seed 0, an 8192-token context, a 4096-token output limit, at most eight calls per task, and a 60-second timeout. The canonical run record stores the exact model digest, chat-template hash, prompt and schema hashes, symbolic-kernel version, dependency versions, hardware metadata, and raw model/tool traces.

Benchmark. The benchmark contains 36 tasks: 12 exact zero-dimensional systems, 8 hidden-solution or witness tasks, 8 structural-diagnosis tasks, and 8 geometry-inspired algebraic systems. The systems contain at most four variables and have maximum degree at most five. Each task record includes its prompt, formal equations, variable order, declared domain, answer mode, task family, and gold answer. Gold records were checked with SymPy and a separately recorded derivation. The suite was frozen before the fresh four-condition rerun; because it also informed earlier pilot development, we treat it as a controlled evaluation suite rather than a pristine held-out benchmark.

Prompts and budgets. No condition uses few-shot examples or task-specific prompt tuning. The task content and final-answer instructions are shared. Condition-specific text only describes the available interaction mechanism: unrestricted Python, the matched helper API, or typed action names and payloads. Tool-enabled conditions use the same maximum-call and timeout policy.

Correctness. Scoring is answer-mode specific. Exact finite tasks require equality with the complete gold solution set up to variable order and exact algebraic equivalence in the declared domain. Witness tasks require at least one valid assignment. Diagnosis tasks require the correct domain-aware structural label. Exact symbolic comparison is preferred; numerical fallback is used only when exact simplification is inconclusive, with residual tolerance 10^{-6}. The scorer version and task-level outputs are frozen in the canonical result record.

Verified correctness. We separately report explicit verification and *verified-correct* answers. Explicit verification records whether the model executed a post-candidate check. Verified correctness is stricter: the final answer must be correct and the retained trace must contain valid evidence covering the claim. For a witness, a claim-linked substitution and domain check suffice. For an all-solutions answer, candidate checks must be accompanied by supported completeness evidence. These labels are produced by the frozen automatic scorer under a condition-independent rubric; because structured traces are easier to parse, we discuss scorer endogeneity separately.

Operational outcomes. A run is marked as containing an *execution error* when its retained trace records a generated-code, helper-invocation, backend, timeout, or tool-execution failure. This is a process metric: a run may contain an error and still recover a correct final answer. The aggregate includes failures in syntax or imports, helper arguments, generated control flow, backend execution, timeouts, and result handling; it must not be read as a count of mathematical solver errors. We also report wall-clock runtime. Failed calls, malformed answers, and unsupported cases remain in the denominator.

Paired analysis. Every condition is evaluated on the same tasks, so the primary comparisons are paired. We report task-level counts and absolute paired differences. The canonical analysis additionally records the 2×2 paired contingency tables, exact McNemar tests, and task-paired bootstrap confidence intervals; subset results are treated as descriptive because of their small sizes.

Reproducibility. All tables are generated from canonical task-level records rather than handwritten aggregates. The companion artifact contains the task and gold manifests, exact prompts, action schemas, shared-kernel implementation, model and environment metadata, raw traces, scoring code, and table-generation commands.

Secondary legacy analyses. To contextualize the fresh controlled comparison, we retain four earlier pilot analyses: a diagnostic-only subset, an oracle-equation presentation that removes language-to-equation ambiguity, and solve-only and check-only restrictions of the typed action surface. These batches predate the canonical shared-helper rerun and have partially documented orchestration differences. We therefore use them only as exploratory mechanism analyses, do not pool them with the fresh batch, and report their full aggregates in Appendix B. Their legacy "evidence" label denotes evidence attachment rather than the stricter verified-correct metric used in the main experiment.

5 Results

Primary shared-helper comparison. With the symbolic helper implementation fixed, the typed interface obtains 25/36 correct answers, compared with 14/36 for matched-helper code, an absolute difference of 11 tasks or 30.6 percentage points. Automatically verified-correct answers increase from 4/36 to 23/36, a difference of 19 tasks or 52.8 points. Thus access to packaged normalization, solving, domain, and checking functions alone does not reproduce the typed condition's performance. The remaining treatment is the deployed interaction format, including constrained invocation, workflow cues, structured statuses, evidence serialization, and the removal of generated glue code; the experiment does not isolate any one of these components or a change in mathematical reasoning.

Packaged helpers are not sufficient. Raw SymPy reaches 17/36 correct answers, whereas matched-helper code reaches 14/36. Simply making the shared kernel importable from generated Python does not improve this model's performance. Instead, matched-helper code contains an execution error in 29/36 runs, compared with 7/36 for raw SymPy and none for the typed interface. This pattern indicates that a capable symbolic API is not sufficient for reliable end-to-end use when the model must synthesize valid invocation code, control flow, and result handling. At the same time, the high matched-helper error rate makes glue-code burden a plausible major source of the correctness

Method	Correct	Verified-correct	Explicit verification	Execution-error runs	Avg. time (s)
Text only	16/36 (44.4%)	0/36 (0.0%)	0/36 (0.0%)	0/36 (0.0%)	7.03
Raw SymPy	17/36 (47.2%)	5/36 (13.9%)	14/36 (38.9%)	7/36 (19.4%)	12.63
Matched-helper code	14/36 (38.9%)	4/36 (11.1%)	4/36 (11.1%)	29/36 (80.6%)	42.98
Typed helper interface	**25/36 (69.4%)**	**23/36 (63.9%)**	**23/36 (63.9%)**	**0/36 (0.0%)**	18.69

Table 1: Fresh four-condition results on the 36-task suite. Verified-correct requires a correct final answer and valid evidence covering the answer-mode claim. Explicit verification records a separate post-candidate check. An execution-error run contains at least one recorded code, helper, backend, timeout, or tool-execution failure; it may still end with a correct answer.

Method	Exact	Hidden	Diagn.	Geom.
Raw SymPy	0/12	**7/8**	8/8	2/8
Matched-helper	0/12	4/8	8/8	2/8
Typed interface	**8/12**	6/8	8/8	**3/8**

Table 2: Correct answers by subset for the three tool-enabled conditions. The small subset sizes support descriptive, not broad population, conclusions.

gap. We therefore treat the result as an interface-usability comparison rather than evidence of improved mathematical reasoning, and we do not relabel every failure as a mathematical solver error.

Correctness and retained evidence. The typed condition produces 23 verified-correct answers, close to its 25 correct answers, whereas raw SymPy and matched-helper code produce only 5 and 4, respectively. The equality between explicit-verification and verified-correct counts in the typed aggregate is empirical, not definitional: a check receives verified-correct credit only when the final answer is correct and the evidence covers the relevant claim. In particular, substitution checks alone do not establish completeness of an all-solutions answer. These numbers therefore measure automatically auditable claim support under the frozen scorer, not the truthfulness of an unrestricted chain of thought.

Where the difference occurs. The largest separation occurs on exact zero-dimensional systems. The typed interface solves 8/12, while both code conditions solve 0/12. These tasks combine enumeration, domain handling, and complete final-answer construction; coordinating those obligations is one plausible source of the observed gap, but the current design does not identify the mechanism. All three tool-enabled conditions solve the eight diagnosis tasks, making that subset a sanity check rather than evidence of an interface advantage. Raw SymPy is strongest on hidden-solution tasks (7/8 versus 6/8 typed), where finding one valid witness can reward flexible exploration. Geometry-inspired systems remain difficult for every condition (3/8 typed versus 2/8 for each code condition).

Runtime and failure behavior. Matched-helper code is both the least accurate tool-enabled condition and the slowest, averaging 42.98 seconds, consistent with repeated generated-code or helper-invocation failures. The typed interface averages 18.69 seconds: slower than raw SymPy, but substantially more accurate and more often verified-correct. Runtime should therefore be interpreted jointly with failure and answer quality rather than as a standalone efficiency ranking.

Secondary mechanism analyses. The legacy pilots provide complementary, non-confirmatory evidence (Appendix B). Supplying equations, variables, and domains explicitly preserved the earlier raw-versus-typed separation (19/36 versus 27/36), suggesting that equation extraction was not its sole source. The diagnostic-only subset was saturated at 8/8 for text, raw code, and typed tools. Within a separate action-surface batch, solve-only reached 22/36 correct, full typed 27/36, and check-only 28/36; check-only also produced the highest legacy evidence-attachment count (31/36), but more error-containing runs than full typed (6/36 versus 1/36). These point estimates motivated the verification-first hypothesis, but different retained batches and the weaker legacy evidence metric prevent pooling or causal interpretation.

Interpretation. The fresh experiment supports a narrow but stronger conclusion than the legacy three-condition comparison. Under one model, one development-informed 36-task suite, one backend, and an identical symbolic kernel, the model-facing interface is associated with substantial changes in correctness, automatically auditable evidence, and execution reliability. The result is not a claim that

type annotations improve mathematical reasoning, that any single interface component causes the difference, or that the typed workflow generalizes beyond this model–benchmark combination.

6 Discussion

The gain persists after matching helper functionality. The matched-helper control resolves the main confound in the original typed-versus-raw comparison. Both matched-helper code and the typed interface invoke the same versioned symbolic kernel, yet the typed condition is 30.6 percentage points more accurate and 52.8 points higher in automatically verified correctness. Access to the same normalization, solving, checking, and domain-handling code therefore does not reproduce the typed condition's performance. This supports a deployed-interface effect under the present model and tasks, but not a claim that the typed condition improves the model's underlying mathematical reasoning.

The treatment is broader than schema syntax. The matched comparison does not identify a single causal primitive. Typed invocation changes argument constraints, workflow guidance, status presentation, error exposure, and evidence serialization together. The paper's claim is consequently about the deployed mathematical interaction format, not about JSON schemas or static types in isolation. Separating these components would require additional factorial interventions, such as neutralizing workflow language while retaining typed calls or giving generated code an equivalent structured response object.

An executable-template control remains open. The matched-helper prompt documents the API, but it does not provide a minimal working program that demonstrates the correct imports and helper invocations. Such a control could reduce low-level code-generation failures while retaining ordinary Python control flow and would help distinguish API scaffolding from the broader structured protocol. Because we did not run this condition, the present study cannot determine whether the typed advantage would remain after that burden is removed. The high matched-helper error rate makes this an especially important target for replication.

A helper API is not automatically usable by an LLM. The matched-helper condition performs worse than raw SymPy and records errors in 29/36 runs. This is a substantive finding: exposing a reliable mathematical library does not ensure that a model can generate correct imports, arguments, control flow, and final-answer serialization. The typed interface removes much of this glue-code burden, which is part of the deployed system rather than a nuisance external to it. However, the magnitude of the error rate also makes baseline usability a central threat to mechanistic interpretation. The aggregate error indicator spans syntax and imports, invalid calls, generated control flow, backend failures, timeouts, and output handling; it should not be read as a count of mathematical solver failures.

Correctness and evidence should be evaluated jointly. A correct answer may be unsupported, while a valid check may cover only one part of an incorrect or incomplete claim. This distinction is especially important for finite solution sets: verifying every returned point establishes candidate validity but not set completeness. The large typed-interface gain in verified correctness suggests that structured traces can make mathematical claims easier to audit, but this metric is partly endogenous because the typed responses are designed for machine parsing. Condition-blind human validation would provide a stronger test of evidence relevance and coverage.

Task structure determines which interface helps. The largest observed separation occurs on exact finite-set tasks, where the model must coordinate enumeration, domain restrictions, completeness status, and final serialization. This pattern suggests, but does not establish, that coordination and serialization burden contribute to the gap. Raw SymPy remains competitive on witness tasks, where one valid point is enough and free-form exploration can be useful. The diagnostic subset is saturated, and the geometry-inspired subset remains difficult. The result is therefore not that structure uniformly improves every mathematical task, but that its value depends on the obligations of the requested claim.

Legacy pilots sharpen, but do not establish, the mechanism. The earlier oracle-equation analysis preserved the typed advantage after the formal system was supplied directly, making language-to-equation extraction an insufficient explanation for that batch. The action-surface pilot found that a check-only interface matched or slightly exceeded the full typed workflow in correctness and legacy evidence attachment, whereas the full workflow

produced fewer error-containing runs. Together these results motivated the present verification-first framing: explicit checking may improve claim support, while a broader action surface may improve operational robustness. Because these pilots used separately retained batches and the old evidence-attachment metric, they are secondary hypothesis-generating analyses rather than confirmation of a specific causal component.

Implications for MathNLP. The study isolates an often hidden layer between mathematical language understanding and symbolic computation: the protocol through which variables, equations, domains, candidates, statuses, and evidence are exchanged. MathNLP evaluations should treat this protocol as an experimental variable rather than attributing all tool-use gains to either the language model or the solver. A broader benchmark should additionally vary linguistic realization—direct equations versus prose, aliases, reordered constraints, implicit domains, coreference, and distractors—to separate language grounding from symbolic interaction.

7 Conclusion

We studied whether the model-facing interface to symbolic algebra matters after controlling for the helper implementation behind it. On 36 polynomial-system tasks, a typed diagnosis–solve–check interface obtains 25/36 correct and 23/36 automatically verified-correct answers. Generated Python calling the same symbolic kernel obtains 14/36 and 4/36, respectively, and contains an execution error in 29/36 runs, compared with none for the typed condition. Raw SymPy and text-only reasoning reach 17/36 and 16/36 correct answers.

Access to the same symbolic functions does not reproduce the typed condition's performance. The typed condition also removes the burden of generating imports, API calls, control flow, and result-handling code, while changing workflow guidance, error reporting, and evidence serialization. The defensible result is therefore an end-to-end deployed-interface effect under one model and a small, development-informed task suite—not evidence that schema syntax improves mathematical reasoning. Whether the advantage remains when generated code is given a minimal executable template is left unresolved.

For mathematical NLP, symbolic functionality and its model-facing protocol should be evaluated separately, and final-answer accuracy should be reported together with claim-aware verification and observable failure behavior.

8 Limitations

Bundled interface effect. The matched-helper control holds the symbolic kernel fixed, but the typed condition still jointly changes schema constraints, workflow guidance, status presentation, evidence serialization, and the need to generate glue code. The results therefore support an end-to-end effect of the deployed interface, not an effect of schema syntax or improved mathematical reasoning in isolation.

No executable-template control. We did not give matched-helper code a minimal working template demonstrating correct imports and helper invocations. Such scaffolding could reduce its high execution-error rate and would provide a cleaner test of the broader structured protocol after low-level API usage is matched. The present comparison cannot predict whether the observed advantage would persist under that control.

Evaluation scope. The study uses one 8B model, one symbolic backend, and 36 small polynomial-system tasks, with subsets of only 8–12 tasks. The benchmark was fixed before the final four-condition evaluation, but had informed earlier development. Subset comparisons are therefore descriptive, and replication with additional models, symbolic systems, independently held-out tasks, and more linguistically diverse problems is needed to establish broader generality.

Errors and verification. Matched-helper code produces execution errors in 29/36 runs. This aggregate combines code-generation, invocation, backend, timeout, and output-handling failures; it is not a mathematical-error taxonomy, and its magnitude limits mechanistic interpretation of the correctness gap. Moreover, candidate substitution establishes validity but not completeness of an ALL-SOLUTIONS claim. Verified correctness follows an automatic, interface-sensitive scorer: structured typed outputs are easier to parse, and no condition-blind human audit was conducted. Unsupported cases receive no completeness credit.

References

Sina Bagheri Nezhad, Yao Li, and Ameeta Agrawal. 2026. SymCode: A neurosymbolic approach to mathematical reasoning via verifiable code generation. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 1489–1503, Rabat, Morocco. Association for Computational Linguistics.

Daniel A Brake, Jonathan D Hauenstein, and Alan C Liddell Jr. 2016. Validating the completeness of the real solution set of a system of polynomial equations. In *Proceedings of the 2016 ACM International Symposium on Symbolic and Algebraic Computation*, pages 143–150.

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

David Cox, John Little, Donal O'shea, and Moss Sweedler. 1997. *Ideals, varieties, and algorithms*. Springer.

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

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

Chengwu Liu, Ye Yuan, Yichun Yin, Yan Xu, Xin Xu, Zaoyu Chen, Yasheng Wang, Lifeng Shang, Qun Liu, and Ming Zhang. 2025. SAFE: Enhancing mathematical reasoning in large language models via retrospective step-aware formal verification. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12171–12186, Vienna, Austria. Association for Computational Linguistics.

Mark Obozov, Michael Diskin, Aleksandr Beznosikov, Alexander Gasnikov, and Serguei Barannikov. 2025. Synthetic proofs with tool-integrated reasoning: Contrastive alignment for LLM mathematics with lean. In *Proceedings of the 3rd Workshop on Mathematical Natural Language Processing*, pages 195–202, Suzhou, China. Association for Computational Linguistics.

Theo Olausson, Alex Gu, Benjamin Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua B. Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176, Singapore. Association for Computational Linguistics.

Ollama. 2026. Ollama. `https://github.com/ollama/ollama`. Version 0.32.0, accessed July 2026.

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

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive APIs. *arXiv preprint arXiv:2305.15334*.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Andrew J Sommese, Charles W Wampler, et al. 2005. *The Numerical solution of systems of polynomials arising in engineering and science*. World Scientific.

Stefan Szeider. 2025. MCP-Solver: Integrating language models with constraint programming systems. *arXiv preprint arXiv:2501.00539*.

Yibo Yan, Jiamin Su, Jianxiang He, Fangteng Fu, Xu Zheng, Yuanhuiyi Lyu, Kun Wang, Shen Wang, Qingsong Wen, and Xuming Hu. 2025. A survey of mathematical reasoning in the era of multimodal large language model: Benchmark, method & challenges. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11798–11827.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

YiFan Zhang, Hanlin Zhang, Li Li, and Eric Xing. 2024. Evaluating step-by-step reasoning through symbolic verification. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2984–3002, Mexico City, Mexico. Association for Computational Linguistics.

A Fresh Shared-Helper Experiment

The confirmatory experiment evaluates the four conditions in Section 4 on the same 36 tasks. The main analysis uses only this fresh batch. Earlier three-condition, oracle-equation, solve-only, and check-only runs motivated the controlled design but are not pooled with the fresh results or used for the paper's primary claims.

A.1 Shared-kernel equivalence

The matched-helper and typed conditions reference the same symbolic-kernel version and helper implementations. The typed actions are thin adapters that validate requests, call the kernel, and serialize its response. The artifact records the helper or kernel hash for each run, generated Python for code conditions, typed requests and responses, normalized equations, variable order, coefficient field, solution domain, and returned execution, candidate, domain, diagnosis, and completeness statuses.

A.2 Exact prompts and budgets

The artifact includes the verbatim system and user messages for all four conditions, the helper API documentation shown to matched-helper code, the exact action schemas shown to the typed condition, and all resolved generation and execution settings. The fresh batch uses temperature 0, seed 0, an 8192-token context, a 4096-token output limit, at most eight calls per task, and a 60-second timeout. Prompt, schema, task, scorer, and kernel hashes are stored in the run manifest.

A.3 Benchmark and gold records

Each task record includes:

- task identifier and family;
- prompt and formal equations;
- variable order, coefficient assumptions, and requested domain;
- answer mode;
- number of variables, maximum degree, and coefficient type;
- gold answer and recorded validation path; and
- construction source and development-exposure metadata.

The 36-task suite was frozen before the fresh rerun, but its prior use in pilot development is explicitly recorded.

A.4 Scoring and evidence obligations

The scorer documents final-answer extraction, exact algebraic equivalence, numerical fallback precision and tolerance, domain validation, diagnosis-label mapping, and the distinction between candidate validity and finite-set completeness. For each task–condition pair, it records final correctness, explicit verification, evidence presence, relevance, validity, coverage, verified correctness, execution status, error category, runtime, and trace identifier. A check invocation alone does not imply verified correctness.

A.5 Paired task-level analysis

All conditions are evaluated on the same tasks. The released task-level table therefore supports the primary matched-helper-versus-typed comparison through its 2×2 paired contingency table, discordant counts, exact McNemar test, absolute paired difference, and task-paired bootstrap interval. The same analysis is provided for raw SymPy versus matched-helper code and raw SymPy versus the typed interface. Subset analyses are reported descriptively.

A.6 Execution-failure taxonomy

The aggregate execution-error indicator is intentionally broad and does not mean that every failure originates in SymPy. The task-level artifact separates at least generated-code syntax, prohibited or failed import, incorrect helper signature or argument, helper exception, backend exception, timeout, unsupported algebra, output-parsing failure, missing final answer, domain mistake, incomplete solution set, and evidence–answer mismatch. Runs with an error remain in the denominator and may still receive answer-correctness credit if they recover a correct final response.

B Legacy Pilot Analyses

The following results recover the exploratory batches that preceded the fresh shared-helper experiment. They are included for transparency and mechanism analysis, but are not pooled with the fresh results. In all tables below, *Evidence* is the original evidence-attachment label—a trace contained residuals, substitution, or structured solver output linked by the legacy scorer—not the stricter verified-correct outcome used in the main paper. Likewise, *Err.* indicates a run containing a tool

or execution error and may overlap with a correct final answer.

B.1 Legacy three-condition comparison

The initial study compared text-only reasoning, unrestricted SymPy code, and the original full typed workflow. It established the descriptive interface gap that motivated the matched-helper control, but bundled helper implementation with invocation format.

B.2 Language-grounding and diagnosis analyses

The oracle presentation supplied equations, variables, and domains explicitly to the tool-enabled conditions. The typed advantage remained descriptive (27/36 versus 19/36), indicating that equation extraction alone did not explain the legacy gap. The diagnostic-only subset was saturated for all conditions and therefore functioned as a sanity check rather than a discriminative benchmark.

B.3 Action-surface analysis

A separately retained batch restricted the typed action surface. Solve-only removed explicit candidate checking, while check-only required the model to propose assignments before verification. Check-only achieved the largest legacy correctness and evidence-attachment point estimates, whereas the full workflow had the fewest error-containing runs. These results motivated the verification-first hypothesis, but the full-workflow row differs from the legacy main batch in explicit-check rate and mean calls; the rows are therefore not interchangeable.

B.4 How the pilots relate to the main experiment

The fresh four-condition study supersedes the legacy main comparison for the paper's central claim because it adds the shared-kernel matched-helper control and uses canonical provenance and stricter verified-correct scoring. The legacy analyses retain a narrower role: the oracle batch tests whether formal-system extraction alone explains the earlier gap, the diagnostic batch identifies a saturated subset, and the action-surface batch motivates explicit verification as a candidate mechanism. None isolates schema constraints, workflow wording, status presentation, or evidence serialization individually.

C Reproducibility Record

The companion artifact contains:

- the exact model identifier, digest, chat template, and reasoning mode;
- Ollama, Python, SymPy, operating-system, and hardware versions;
- decoding, context, token, call, retry, and timeout settings;
- canonical task, gold, run-config, tool-call, run, and score records;
- raw model messages, generated code, typed requests, and tool traces;
- the frozen scorer and table-generation code; and
- environment files and commands that regenerate every reported table.

Method	Correct	Evidence	Explicit check	Error runs	Audit abst.	Calls	Time (s)
Text only	15/36	0/36	0/36	0/36	21/36	0.00	8.76
Raw SymPy	17/36	7/36	17/36	9/36	18/36	1.69	12.53
Full typed workflow	27/36	27/36	10/36	1/36	9/36	2.03	10.98

Table 3: Legacy main-batch aggregates. This comparison predates the matched-helper control and therefore does not isolate the deployed interaction format from packaged helper functionality.

Setting	Method	Correct	Evidence	Explicit	Err.	Abst.	Calls	Time (s)
Diagnostic (8)	Text only	8/8	0/8	0/8	0/8	0/8	0.00	8.02
Diagnostic (8)	Raw SymPy	8/8	0/8	0/8	2/8	0/8	1.25	7.87
Diagnostic (8)	Typed interface	8/8	1/8	0/8	1/8	0/8	1.12	6.44
Oracle (36)	Raw SymPy	19/36	9/36	20/36	12/36	17/36	1.81	15.41
Oracle (36)	Typed interface	27/36	27/36	12/36	5/36	8/36	2.25	19.44

Table 4: Legacy diagnostic-only and oracle-equation analyses. The oracle batch reduces language-grounding ambiguity but still bundles helper functionality and interface format.

Action surface	Correct	Evidence	Explicit check	Err.	Abst.	Calls	Time (s)
Full typed workflow	27/36	27/36	6/36	1/36	9/36	1.86	10.96
Solve-only	22/36	27/36	0/36	5/36	14/36	1.00	7.38
Check-only	28/36	31/36	36/36	6/36	8/36	1.75	8.95

Table 5: Legacy action-surface pilot. Point estimates are descriptive and use the legacy evidence-attachment metric. They are not pooled with the fresh shared-helper experiment.

Hard or Just Unreached? Diagnosing the Sampling Blind Spot in Math-Reasoning Difficulty Estimation

Luca Zhou[1], **Sajel Shah**[2], **Emanuele Rodolà**[1,3], **Roberto Dessì**[2]
[1]Sapienza University of Rome, [2]Not Diamond, [3]Paradigma
Correspondence: luca.zhou@uniroma1.it

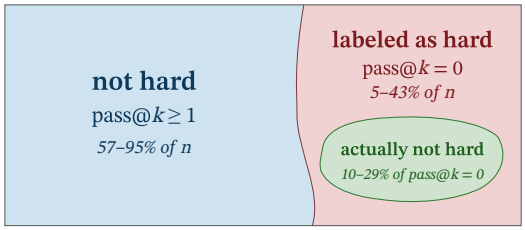


Figure 1: *The sampling blind spot (n=1000).* Across twelve (model, benchmark) cells, pass@k flags 5–43% of items as "hard" (red, pass@$k = 0$). A compute-matched deterministic regime (six greedy chains with residual-stream interventions) reaches 10–29% of that stratum (green), showing that a non-trivial fraction of items pass@k calls "hard" are structurally identifiable in the model's residual stream rather than truly hard.

Abstract

Math and science reasoning benchmarks rely on pass@k, the fraction of sampled chains that reach gold, as the canonical per-example difficulty signal. We show this proxy has a persistent blind spot on its hardest stratum: on the eight free-form math cells we test (GSM8K and MATH across four open-weight models), 10.3–22.9% of the examples that no sampling seed solves in six tries are instead solved at matched compute by a six-chain deterministic regime. These are greedy decoding plus five cheap residual-stream perturbations applied via activation grafting, while greedy alone solves at most 6% on these math cells. Recovery scales with the additional budget, across perturbations whose mechanistic distinctness we verify across all twelve cells (cross-kind fix-set Jaccard ≤ 0.47 in every setup. Activation grafting is used as an intervention on internal representations, not a decoding method; we use it purely as a diagnostic and diversification tool, and our recovered items show that the pass@$k=0$ stratum is structurally identifiable in the residual stream rather than that the unmodified model reaches them under ordinary inference.

1 Introduction

RL pipelines with verifiable rewards, math/code data curation, difficulty-stratified curricula, and pass@k-style hardness annotations all share a single label-required primitive: examples on which no sampled chain reaches gold are filtered out, down-weighted, or labelled hard (Lambert et al.; Guo et al., 2025; Shao et al., 2024; Yuan et al., 2023; Toshniwal et al., 2024; Zhou et al., 2026). The signal is the pass@k count itself, the number of sampled chains that reach gold (Cobbe et al., 2021; Chen et al., 2021); where a single answer is needed, k chains are aggregated by majority vote or self-evaluation (Wang et al., 2022; Chen et al., 2023; Aggarwal et al., 2023). The first sample, the second, the kth are all drawn the same way, and temperature sampling is treated as *the* source of decoding diversity. A problem that no sampling chain solves is treated, by some reasonable proxy, as difficult.

We show this last assumption fails on a structurally identifiable slice. Through three empirical observations across four open-weight instruction-tuned models (`Qwen-2.5-3B`, `Llama-3.2-3B`, `Llama-3.1-8B`, `Mistral-Nemo-12B`) and three reasoning benchmarks (GSM8K, MATH, MMLU-Pro), we argue that sampling has a *persistent blind spot* on its hardest stratum at the budgets we test, and that a non-trivial fraction of the slice flagged as hardest by sampling-only estimators is in fact reachable by deterministic decoding at matched compute.

(1) Greedy is competitive at a single sample. Across the (model, benchmark) cells we test, greedy decoding is competitive with or better than the mean single-sample accuracy, with the gap reaching up to 2.8%. The default single-pass baseline for self-consistency is therefore already weaker than a deterministic alternative.

(2) Sampling has a persistent blind spot on its hardest stratum. The pass@$k = 0$ stratum,

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 13–29
October 29, 2026 ©2026 Association for Computational Linguistics

examples no sample reaches gold in k tries, is the operative signal for any pipeline whose deployment already has labels: RL with verifiable rewards, math data curation, and pass@k-style hardness annotation, etc. At $k = 6$ this stratum is large on free-form math: 5.1–8.3% of GSM8K and 28.7–43.5% of MATH across the four models we test (51–435 examples per setup; Table 3). The per-sample marginal-shrink rate decreases monotonically through our sampling budget (Fig. 3; App. C), so the stratum is not an artifact of an undersized k. This stratum is treated, by some reasonable proxy, as the slice the model cannot solve. The question we focus on: is that slice intrinsically hard, or is it reachable but only off the stochastic axis? See the intuition in Fig. 1.

(3) The deterministic regime reaches what six samples miss. We isolate the slice on which sampling truly fails, examples no sampling seed reaches in six tries, and ask how much of it a deterministic regime at matched compute reaches. Activation grafting is the diagnostic: replacing the last prompt-token hidden state with cheap synthetic vectors, then decoding greedily, holds prompt and weights fixed while perturbing only the deterministic trajectory. A six-chain deterministic regime (greedy plus five grafts) reaches 10.3–22.9% of these examples on the eight free-form math cells (GSM8K, MATH; 10–29% across all twelve cells), while greedy alone reaches only up to 9% (Sec. 5). The recovery scales with deterministic budget across five graft kinds whose mechanistic distinctness, we verify across all twelve cells (Sec. 5, App. H: cross-kind fix-set Jaccard ≤ 0.47 in every cell, with the single same-axis pair of random probes reaching 0.76 on `Qwen-2.5-3B` / GSM8K as an illustrative ceiling). This rules out both the "intrinsically hard" and the "one privileged vector" readings. The point is not that deterministic beats sampling on aggregate accuracy; it is that the two regimes have partially disjoint correct sets, and the slice flagged as hardest by sampling-only proxies might not be the actual hardest slice.

The logical chain and its implication. Sampling has a persistent blind spot on its hardest stratum, and the deterministic axis reaches into it: recovery on the no-sample-reaches-gold stratum scales with the deterministic budget (one chain → six chains: 0–9% → 10–29%) across mechanistically distinct grafts. So on the (model, benchmark) cells we test, sampling-derived difficulty estimates (pass@k filters, RL data-curation pipelines, hardness annotations, curricula) mislabel a structurally identifiable 10–29% of their hardest stratum, conflating "hard" with "unreached on the stochastic axis." Grafting is not a decoding method and we do not propose to deploy it; we use it purely as a diagnostic to locate this slice, at the cost of one extra deterministic forward pass per example plus a few cheap perturbations.

Why this matters for mathematical reasoning. Math reasoning is where pass@k-style difficulty labels are most consequential and where our effect is largest in absolute terms: a structurally identifiable 10.3–22.9% of the "hardest" GSM8K / MATH bucket is reached by an edited forward pass of the same model at matched compute, so the bucket is partly an artifact of the decoding regime, not of the items. This directly affects benchmark hardness buckets, difficulty-calibrated curricula and per-example weights, rejection-sampling data curation (Yuan et al., 2023; Toshniwal et al., 2024), and verifier / reward-model datasets, all of which inherit the same blind spot when built from sampled-chain correctness alone.

2 Related Work

Self-consistency and inference-time scaling. Self-consistency aggregates k sampled chains by majority vote (Wang et al., 2022) and is the default protocol for inference-time scaling on reasoning benchmarks (Wang et al., 2022; Chen et al., 2023; Aggarwal et al., 2023). Variants adaptively allocate the budget (Aggarwal et al., 2023), weight chains by self-evaluation (Chen et al., 2023), or combine sampling with verifier-guided search (Cobbe et al., 2021; Lightman et al., 2024). Across this line of work, $k = 1$ greedy is reported as a weak baseline that majority-voted sampling improves on. We do not dispute the aggregate ranking; we point out that the cell where greedy is correct and sampling-majority is not is non-trivial and persistent, and is hidden by aggregate accuracy.

Decoding diversity. Existing methods for decoding diversity operate in token space: nucleus and top-k sampling (Holtzman et al., 2020; Fan et al., 2018), diverse beam search (Vijayakumar et al., 2016; Kool et al., 2019), and contrastive decoding (Li et al., 2023); see Welleck et al. for a survey of inference-time decoding. All inject diversity through the per-step output distribution. Our diagnostic is orthogonal: the pass@$k = 0$ stratum persists at $k = 6$ under standard self-consistency

hyperparameters, and is reached by deterministic perturbations applied to the residual stream rather than the output distribution.

Hidden-state and representation interventions. A growing body of work edits or analyzes internal activations: causal tracing for factual recall (Meng et al., 2022), representation engineering and steering (Zou et al., 2023; Turner et al., 2024; Rimsky et al., 2024), and cross-model communication via projected hidden states (Ramesh and Li, 2025). We borrow this primitive only as a probe, a minimal residual-stream perturbation whose deterministic outputs we compare against sampling, not as a method that should be deployed at inference.

Test-time compute and difficulty estimation. Recent work on test-time compute scaling reports favourable exchange rates between extra inference passes and accuracy on reasoning benchmarks (Snell et al., 2025; Brown et al., 2024; Wu et al., 2025). A common companion measure scores per-example difficulty by pass@k or sampling agreement (Cobbe et al., 2021; Brown et al., 2024), and the same signal is used to filter hard subsets out of a larger dataset (Zhou et al., 2026). Downstream RL-with-verifiable-rewards pipelines (Lambert et al.; Guo et al., 2025) consume this signal: prompts on which the model never samples a correct answer contribute no reward gradient. Our second observation directly affects this practice: a non-trivial fraction of examples that fail under sampling are solved by deterministic decoding, so sampling-only difficulty estimates conflate "hard" with "unreachable on the stochastic axis". Prior work on inference-time scaling has measured how coverage of correct answers grows with the sampling budget (Brown et al., 2024), and complementary theoretical analyses (Levi, 2025, 2026) characterise pass@k as exhibiting a power-law decay in k whose exponent depends on the interaction between per-example difficulty and training; Levi (2026) further shows that learning "shrinks the hard tail" of the failure distribution until an irreducible target-difficulty limit is reached. Our angle is orthogonal: rather than modelling how the pass@k failure rate shrinks with k under sampling, we ask what the residual pass@$k=0$ stratum *actually contains*, and isolate a structurally identifiable slice of it that a matched-compute deterministic regime reaches while the sampling axis does not.

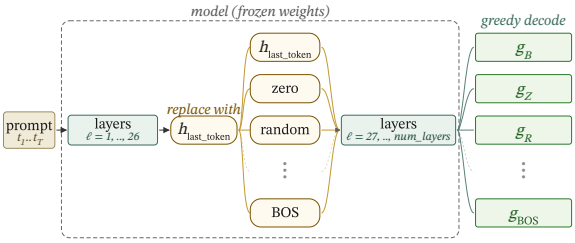


Figure 2: *Constructing the deterministic regime.* The frozen model is run on a prompt up to layer 26, producing the last-prompt-token residual-stream activation $h_{\text{last\_token}}$. It is then *replaced* with one of k cheap, parameter-free alternatives: $h_{\text{last\_token}}$ itself (the greedy baseline), the zero vector, a Gaussian sample, ..., the BOS-token activation. The same model continues its forward pass through layers 27 to `num_layers` under greedy decoding. The k chains differ only in this one residual vector, yielding k deterministic outputs $g_B, g_Z, g_R, \ldots, g_{\text{BOS}}$.

3 Setup

We evaluate on the cross product of four open-weight instruction-tuned models, `Qwen-2.5-3B` (Qwen Team, 2024), `Llama-3.2-3B`, `Llama-3.1-8B` (Llama Team, 2024), and `Mistral-Nemo-12B` (Mistral AI Team, 2024), and three reasoning benchmarks, GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021) and MMLU-Pro (Wang et al., 2024); $n = 1000$ matched prompts per cell, and `max_new_tokens`$= 2048$ to prevent truncation.

Decoding regimes. The *sampling regime* draws six chains $S_{1...6}$ at $T = 0.7$, $p_{\text{top}} = 0.9$; we report sample-level quantities (pass@k, single-sample mean) directly, and where a sample aggregate is needed we use majority vote with ties broken uniformly at random over 50 seeds. A temperature sweep at $T \in \{0.3, 0.7\}$ appears in §8 (Table 5). The *deterministic regime* consists of greedy chains that differ only in a single last-prompt-token activation graft at $\ell = 26$, fired during prefill only; decoding is greedy throughout. All graft vectors are fixed across the dataset (no per-example optimisation, parameter-free, no extra forward passes beyond the prefill hook). Figure 2 provides an overview of grafting, while table 1 lists the eight chains used in the paper.

Matched compute. By *matched compute* we mean k deterministic chains against k sampled chains: the same number of prefill+decode forward passes per example. The graft is a sin-

gle `register_forward_hook` that overwrites one residual-stream vector during prefill (no extra forward pass, no gradient, negligible wall-clock overhead), so we operationalise "compute" as the FLOP count of those k passes. The headline claim of Sec. 5 is at $k=6$.

Table 1: Deterministic chains used in the paper. Short names are used interchangeably with the $g_\cdot$ symbols from §5 onward.

Symbol / name	Description	Set
g_B	greedy, no graft	3
g_Z / ZERO	zero vector	3
g_R / RAND	norm-matched Gaussian	3
RUNIT	random unit direction	6
SHUF	hidden-dim permutation of activation	6
BOS	BOS-token activation	6
AVG	mean over prompt-token activations	8
PREV	same position from $\ell-1$	8

Chains were chosen to be mechanistically distinct in residual-stream geometry (the origin, a high-norm random direction, a token-position substitution, a hidden-dim permutation, a previous-layer state). We quantify the resulting fix-set diversity in §5 (cross-kind Jaccard ≤ 0.47 in every cell vs. a same-axis ceiling reaching 0.76 between two random-direction probes; App. H). Layer and position choices shift the oracle ceiling by ≤ 4pp and reorder leading methods (Apps. B, J, H); the existence of the recovered slice is insensitive to these choices, the specific chain that recovers a given example is not.

4 Greedy is Competitive at $k=1$

At $k=1$, the sampling regime is one stochastic draw S_i, the deterministic regime is g_B. Running six independent sampling seeds (42–47), greedy is competitive with or better than the mean single-sample accuracy (Table 2), with the gap reaching up to $+2.8\%$ on `Llama-3.2-3B` / MATH. On six of twelve cells it exceeds the seed-to-seed range and beats the best seed. Greedy follows the local MAP trajectory while a single sample is a high-variance estimator of it; on harder problems the per-step distribution is flatter, and MAP becomes a better single-pass bet than any draw.

The deterministic regime's clearest per-pass advantage is at $k=1$; at higher k, its contribution is better understood in terms of coverage (Sec. 5).

Table 2: Single-pass head-to-head over six sampling seeds.

Setup	$1S$ Mean [Min,Max]	g_B	Δ
Qwen-2.5-3B/GSM8K	0.84 [0.84, 0.85]	0.84	+0.00
Qwen-2.5-3B/MATH	0.54 [0.54, 0.55]	0.56	+0.01
Qwen-2.5-3B/MMLU-Pro	0.40 [0.40, 0.41]	0.40	−0.00
Llama-3.2-3B/GSM8K	0.75 [0.74, 0.76]	0.77	+0.02
Llama-3.2-3B/MATH	0.46 [0.45, 0.47]	0.49	+0.03
Llama-3.2-3B/MMLU-Pro	0.35 [0.34, 0.36]	0.37	+0.02
Llama-3.1-8B/GSM8K	0.84 [0.83, 0.85]	0.85	+0.01
Llama-3.1-8B/MATH	0.49 [0.48, 0.49]	0.49	+0.00
Llama-3.1-8B/MMLU-Pro	0.48 [0.46, 0.50]	0.50	+0.01
Mistral-Nemo-12B/GSM8K	0.85 [0.84, 0.85]	0.86	+0.01
Mistral-Nemo-12B/MATH	0.32 [0.31, 0.34]	0.34	+0.02
Mistral-Nemo-12B/MMLU-Pro	0.43 [0.42, 0.44]	0.44	+0.01

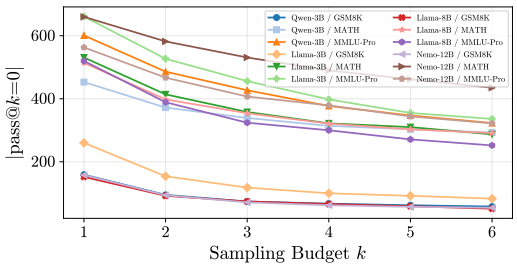


Figure 3: Size of the pass@$k=0$ stratum as a function of sampling budget k, across the twelve (model, benchmark) cells ($n=1000$ each). The curve flattens early: most of the shrink from $k=1$ to $k=6$ happens by $k=3$, and a residual 5.1–43.5% of prompts remains unreached at $k=6$. The diminishing returns motivate spending budget on a different axis.

5 The Blind Spot is Structural, Not Hardness

The stratum. The pass@k=0 stratum, examples no sampled chain in k reaches gold, is the operative signal for label-required pipelines (RL with verifiable rewards, data curation, hardness annotation). At $k=6$ this stratum holds 5.1–43.5% of prompts (51 to 435 examples per setup; see Table 3). This is the slice Sec. 5 attacks at matched compute.

The stratum persists at higher k. The per-sample marginal-shrink rate of pass@$k=0$ decreases monotonically across all twelve setups (Fig. 3; App. C). Within our sampling budget the shrinkage has already plateaued: pass@$6=0$ still contains 5.1–43.5% of prompts, and additional samples do not reliably close the gap. The stratum we attack is therefore not an artifact of an undersized sampling denominator.

Two readings of the residual slice are possible: those examples are intrinsically hard, or they are reachable but not on the stochastic axis. Section 5

adjudicates between them.

Anatomy. The recovered pass@6 = 0 examples share a recurring structure: the prompt admits two internally coherent readings of a discrete choice, sampling drifts toward one reading on all six seeds, and at least one deterministic chain commits to the gold reading. Which chain rescues a given example varies by example: no single graft dominates, but the union over a handful of mechanistically distinct chains catches a non-trivial fraction of the stratum. As a representative case, MATH Counting & Probability `0445` (`Qwen-2.5-3B`, gold 588) is missed by all six sampled seeds in $\{42, \ldots, 47\}$ yet recovered by two of the three deterministic chains, which correctly account for the identical-colour multiplicities and circular symmetry while the third collapses to a distinct-object count. This case and two further worked examples, spanning all three rescuing pairs from $\{g_B, g_Z, g_R\}$, are detailed in App. G.

Adjudicating the two readings. To adjudicate between the two readings of the residual sampling-failure slice, we work directly on the pass@6=0 stratum: examples for which no seed in $\{42, \ldots, 47\}$ reaches gold. On this slice, no sampled chain in six tries was correct: this is the strictest sampling-failure test. The slice contains 51 to 435 examples per setup (Table 3). We then ask: of these, how many does a deterministic regime at matched compute reach? Those would not actually be hard, but anyway flagged as hard by a sampling-based filter.

We define R_k as the number of pass@6=0 examples on which at least one of k deterministic chains is correct. R_1 is greedy alone. R_3 adds zero and norm-matched random grafts (g_Z, g_R). R_6 extends to greedy plus five grafts (zero, random, random unit direction, hidden-dim permutation, BOS-token replacement), matching the $k = 6$ sampling budget. R_8 adds mean-activation and previous-layer grafts. All grafts are at $\ell = 26$, last token position. The headline matched-compute claim is R_6 vs. pass@6=0 (six deterministic chains against six sampled chains, on the stratum where no sample reaches gold); R_1 and R_3 are reported at sub-matched budget and so strengthen, not weaken, the comparison (one or three deterministic chains reaching examples that six samples miss).

Three observations follow. (a) At matched compute, R_6 recovers 10–29% of examples that six sampling seeds collectively miss. The pass@6=0 stratum is therefore not intrinsically hard: a non-trivial fraction is reachable by deterministic decoding given enough representatives. (b) R_1 is still relatively small (0–9%), so a single deterministic chain is closer to "another sample" than to a separate regime; the regime emerges once multiple mutually disjoint perturbations are admitted. The recovery scales near-linearly with the deterministic budget (R_6 is several times R_1 on every setup where $R_1 > 0$). (c) The recovery is not driven by one privileged vector. Among the seven grafts surveyed, no single graft kind is the single largest individual contributor to R_6 on more than half the cells, and the leader rotates across cells (SHUFFLED on the math-style cells we inspected, RANDOM on `Mistral-Nemo-12B` / MMLU-Pro, multi-way ties on the smallest strata). Across all twelve cells, pairwise cross-kind fix-set Jaccard never exceeds 0.47 (range 0.06–0.47; App. H, Table 7), while the single same-axis pair (random vs. random-unit, two norm-matched Gaussians) reaches 0.76 on `Qwen-2.5-3B` / GSM8K as an illustrative ceiling for what resampling along a single random axis can contribute (see App. H for the per-cell breakdown). Different graft kinds therefore probe different subsets of the stratum in every cell we test, which is consistent with the recovery scaling with the deterministic budget.

These three readings together rule out both the "intrinsically hard" explanation and the "one privileged vector" explanation. Crucially, the recovery on pass@6=0 *scales with the deterministic budget*, admitting more mutually disjoint deterministic perturbations reaches more examples no sampling seed reached, and the dimensions on which it scales are mechanistically distinct from each other and from sampling's stochastic axis. What looked intrinsically hard under sampling was, for a non-trivial fraction, just unreached on the stochastic axis: the model was close, the missing ingredient was a different *kind* of diversity.

66–88% of pass@6=0 remains unadjudicated by an 8-chain deterministic regime: those examples may be intrinsically hard, or reachable on some other diversity axis we do not probe. Within the 3-chain set, each of $\{g_B, g_Z, g_R\}$ contributes some examples uniquely (Table 6): no single chain dominates across the twelve cells, and the deterministic budget pays back per chain. Per-setup unique contributions sit in the 1.3–8.9% band of $n = 1000$, with g_B leading in nine of twelve cells but never to the point of making either graft redundant, remov-

Table 3: Deterministic recovery on the pass@6=0 slice. R_k counts examples where no sampling seed in $\{42, \ldots, 47\}$ reaches gold but at least one of k deterministic chains does. Percentages are of the pass@6=0 slice (column 2); $n = 1000$ for all cells.

Setup	p@6=0	R_1	R_3	R_6	R_8
`Qwen-2.5-3B` / GSM8K	58	0 (0.0%)	5 (8.6%)	8 (13.8%)	10 (17.2%)
`Qwen-2.5-3B` / MATH	293	10 (3.4%)	23 (7.8%)	39 (13.3%)	48 (16.4%)
`Qwen-2.5-3B` / MMLU-Pro	323	14 (4.3%)	43 (13.3%)	81 (25.1%)	100 (31.0%)
`Llama-3.2-3B` / GSM8K	83	2 (2.4%)	12 (14.5%)	19 (22.9%)	21 (25.3%)
`Llama-3.2-3B` / MATH	287	11 (3.8%)	24 (8.4%)	42 (14.6%)	50 (17.4%)
`Llama-3.2-3B` / MMLU-Pro	336	30 (8.9%)	66 (19.6%)	96 (28.6%)	114 (33.9%)
`Llama-3.1-8B` / GSM8K	51	3 (5.9%)	7 (13.7%)	10 (19.6%)	11 (21.6%)
`Llama-3.1-8B` / MATH	291	9 (3.1%)	22 (7.6%)	30 (10.3%)	36 (12.4%)
`Llama-3.1-8B` / MMLU-Pro	252	19 (7.5%)	46 (18.3%)	72 (28.6%)	80 (31.7%)
`Mistral-Nemo-12B` / GSM8K	54	1 (1.9%)	8 (14.8%)	11 (20.4%)	13 (24.1%)
`Mistral-Nemo-12B` / MATH	435	16 (3.7%)	39 (9.0%)	64 (14.7%)	75 (17.2%)
`Mistral-Nemo-12B` / MMLU-Pro	322	11 (3.4%)	55 (17.1%)	93 (28.9%)	109 (33.9%)

ing any one chain drops the 3-chain oracle by at least 13 examples on every setup, and the leading and trailing contributor differ by a factor of ≤ 2.2 (per-cell breakdown in Table 6, App. F).

6 Mechanism: Residual-Stream Perturbation, Not Attention Rerouting

Counts and case studies establish that grafts reach examples sampling does not. They do not, on their own, explain why the deterministic chains in our table behave as multiple distinct axes rather than a single noisy one. This section gives the answer.

We instrumented `Qwen-2.5-3B` with greedy attention on 100 GSM8K examples and ran each twice: once unperturbed, once with a last-token graft at $\ell = 26$ (random and zero, separately). For every transformer block we recorded (i) the hidden state at the graft position before and after grafting, yielding cosine similarity and L_2 distance per layer, and (ii) the full attention-weight matrices, allowing direct comparison between baseline and grafted runs at identical token positions. For the random graft, $4/100$ examples exhibited a runaway-divergence mode in which L_2 grows to 10^4–10^5 in deeper layers; we exclude these from the per-layer summary to avoid heavy-tail contamination and report the trajectory on the remaining 96. Zero shows no such outliers.

Hidden-state divergence is shown in Figure 4 (per-layer numbers in App. A, Table 4). At $\ell = 26$, the random graft makes the hidden state essentially orthogonal to the baseline (cos $= 0.001$) at L_2 distance 121.8; the zero graft sets it to the origin ($L_2 = 86.2$, the baseline norm itself). Crucially the divergence does *not* decay through the remaining nine layers: L_2 grows monotonically to 466.7 (random) and 415.1 (zero) by the final layer, while cosine similarity to baseline remains below 0.37. In this model, the forward computation does not project the perturbation back onto the baseline trajectory; it carries it forward.

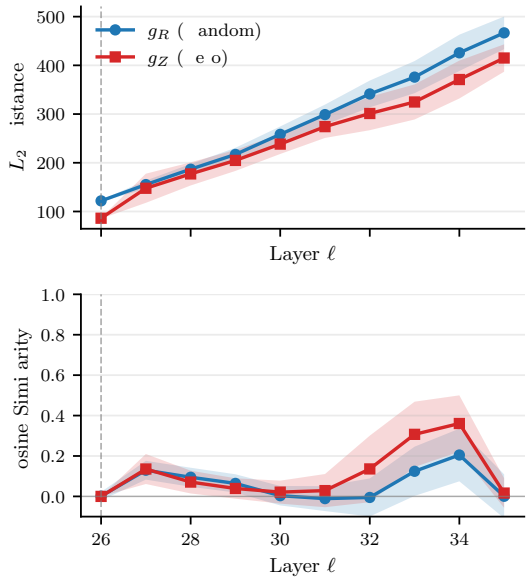


Figure 4: Hidden-state divergence at the graft position over layers $\ell \geq 26$ (`Qwen-2.5-3B`, GSM8K, last token, $\ell = 26$ graft; random excludes 4 runaway-divergence outliers, $n = 96$). Cosine similarity to the unperturbed run stays low and L_2 distance grows monotonically through the remaining nine blocks: the perturbation is carried forward, not absorbed.

By contrast, the graft's *direct* effect on attention is negligible (attention over the divergent generation itself is not what we measure here; see footnote). We report two quantities: (a) the mean abso-

lute difference in `prefill_attn_received` (per-token attention received during the prompt forward pass, averaged over heads and query positions) between baseline and grafted runs, and (b) the mean absolute difference in the full attention-weight matrix under teacher forcing on *identical* token sequences (prompt + baseline output for both runs, so any difference is attributable to the graft and not to divergent generation).[1] On the 100 examples, (a) $= (1.1 \pm 0.3) \times 10^{-5}$ and (b) $= (3.5 \pm 0.8) \times 10^{-5}$ for the random graft; the zero graft is within rounding. This is four to five orders of magnitude smaller than the residual-stream perturbation the same intervention induces.

From mechanism to behaviour. The two measurements above pin down how the graft acts on the forward pass: it does not reroute attention but injects a content vector that the remaining nine blocks carry forward without correction (Figure 4, L_2 grows monotonically; attention deltas are 10^{-5}). Mechanistically, each graft vector is therefore a different initial condition for the same nine-block computation producing the first generated token, and because the forward pass does not contract these perturbations back onto the baseline trajectory, distinct graft kinds produce distinct downstream trajectories that the autoregressive decode then locks in. This is the link to the behavioural table: different residual-stream perturbations propagate into different generated answers, and the eight chains in Table 3 are eight non-collinear initial conditions, not eight redraws from one.

The deterministic axis is therefore multi-dimensional in a behaviourally measurable sense: cross-kind fix-set Jaccard ≤ 0.47 in every one of the twelve cells (App. H) shows the chains act on largely disjoint subsets of the input distribution. That is why R_k scales with the deterministic budget rather than saturating after one graft: more graft kinds are more independent initial conditions, each reaching its own slice of pass@6=0.

[1] Teacher forcing the grafted run on the baseline output is necessary to isolate the graft's direct effect on attention from indirect effects mediated by divergent generation: the grafted chain typically produces a different output, and attention over that different output would conflate the two. Comparing on the *same* token sequence asks the cleaner question: does the graft, at layer $\ell = 26$, change which tokens subsequent attention heads attend to?

7 Practical Utility

The headline finding of §5 is qualitative: a six-chain deterministic regime reaches 10–29% of pass@6 $= 0$. This section turns it into two *deployable* recipes that need neither gold labels nor per-cell tuning. First, a matched-cost substitution: a fixed-policy rule that spends one of B forward passes on the mean-prompt-activation graft AVG rather than an extra sample covers more of the dataset than B samples on the majority of (cell, budget) combinations, with AVG chosen once and never tuned. Second, a label-free curation flag: chain disagreement among $\{g_B, S_0, S_1\}$ ranks recoverable items above the base rate without gold labels, with 3–5$\times$ lift on free-form math at three forward passes per item. Both recipes are deployable on top of an existing pass@k curation pipeline, at the cost of a small constant per-item overhead.

7.1 Matched-cost coverage

A sharper test of the §5 finding is direct substitution at matched compute: at total budget B forward passes per item, does $(B-1)$ samples plus *one* deterministic chain cover more of the dataset than B samples? We propose the mean-prompt-activation graft AVG as a fixed, no-tuning rule: at every budget and every cell, spend one of the B forwards on AVG rather than an extra sample. Across the 96 (cell, B) slots with $B \in \{2, \ldots, 9\}$, the fixed-AVG policy reduces residual pass@$B = 0$ count by $+1.66$ items per slot on average, and is net-positive on 8 of 12 cells: all four MMLU-Pro cells (mean Δ $+4.4$ to $+7.0$ per slot), `Qwen-2.5-3B` / MATH ($+7.9$), `Mistral-Nemo-12B` / MATH ($+4.5$), `Llama-3.1-8B` / GSM8K ($+2.0$), and `Mistral-Nemo-12B` / GSM8K ($+0.4$). As an upper bound, we also report the oracle in which the best chain at each (cell, B) is picked from the eight in Table 1; this is an analysis, not a deployable rule. The oracle wins on 80 of 96 slots and is particularly large on MMLU-Pro and Mistral-Nemo cells (e.g. `Mistral-Nemo-12B` / MMLU-Pro: $379 \to 357$ at $B = 4$ and $277 \to 269$ at $B = 9$; `Llama-3.2-3B` / MMLU-Pro: $336 \to 317$ at $B = 6$; `Qwen-2.5-3B` / MATH: -10 at $B = 6$). The unconstrained best mix even goes det-heavy on three cells at $B = 9$ ($s = 1, d = 8$ on `Qwen-2.5-3B` / MATH; $s = 2, d = 7$ on `Mistral-Nemo-12B` / MMLU-Pro; $s = 4, d = 5$ on `Llama-3.2-3B` / GSM8K). The gap between fixed-AVG and the oracle is modest at the slot level (median best-chain wins $\approx$ 1–2 items more than

AVG when they differ), so AVG reaches near-oracle coverage with no per-cell tuning. Per-cell, per-B oracle tables are in App. K. The headline is operational: when budgeting forward passes for coverage on a curation task, spending one of them on the AVG graft rather than an extra sample is the better choice on the majority of (cell, budget) combinations. The choice of a *single* graft slot is a proof-of-concept simplification: the same matched-cost substitution extends to $(B-d)$ samples plus d deterministic chains for any $d \leq B$, and a curation pipeline with a fixed audit budget per item can tune d to its own sample/deterministic exchange rate.

7.2 Toward label-free identification

A natural follow-up is whether the recoverable slice can be flagged *without* gold labels. As a deployable probe we use three forward passes per item, the greedy chain g_B and two sampling seeds S_0, S_1, and rank items by how many of the three pairs disagree. Against a 0.8–9.6% base rate of recoverable items, the probe concentrates them at the top of the ranking on the eight free-form math cells (lift 3–5$\times$ at $K = 2$–5%; precision up to 20% at $K = 1\%$ on `Llama-3.1-8B` / MATH), but the signal largely vanishes on the bounded-vocabulary MMLU-Pro cells (lift 1.0–1.5$\times$), so the "3–5$\times$" headline is a majority claim, not universal. This reframes the deployment as a label-free curation tool, at three forward passes and no gold labels it flags the $K\%$ of items most likely to be *mislabelled* as intrinsically hard; a calibrated precision guarantee would require a small labelled dev set, which we leave to future work. Per-setup numbers and precision–recall curves are in App. I (Table 9, Fig. 6).

8 Discussion

Robustness. Mechanism is residual-stream perturbation, not attention rerouting (§6), and the diagnostic strengthens at lower sampling temperature rather than washing out: on `Qwen-2.5-3B` / GSM8K at $T = 0.3$ the pass@4 $= 0$ stratum *grows* and R_6 on it rises from 20.9% to 31.7%, so the blind spot is not a high-temperature artifact (App. D, Table 5).

Intervention vs decoding. Activation grafting is an intervention on representations, not a decoding method: it overwrites the layer-26 activation with a fixed vector and reads predictions off the edited forward pass. Our claim is correspondingly narrow: of the examples pass@$k = 0$ classifies as too hard, a non-trivial fraction is reached by an edited forward pass at matched compute, so the slice is *structurally identifiable* in the residual stream. We do not claim those items are "truly easy" or that the unmodified model reaches them under ordinary inference — only that their reachability is decoding-regime dependent rather than intrinsic, and that there is no difficulty oracle: any operational notion of difficulty (pass@k, sampling agreement, intervention-based reachability) is a choice, not a measurement. The reverse asymmetry is real but unquantified: some items the deterministic regime misses are reachable by sampling. We do not probe that direction because downstream pipelines do not act on it, difficulty estimation defaults to sampling, and the deterministic regime is collapsed to greedy alone (on which a pass@k signal is undefined), so what needs auditing is the direction we report: items the sampling-based proxy flags as hard.

9 Conclusion

Pass@k is the canonical sampling-based primitive for per-example difficulty, driving RL data curation, hardness annotation, and curricula across reasoning-LM evaluation. On the twelve (model, benchmark) cells we test (four open-weight models spanning 3B, 8B and 12B parameters across three reasoning benchmarks), we show it has a persistent blind spot on its hardest stratum at the $k = 6$ budget: of examples no sampling seed in $\{42, \ldots, 47\}$ solves, 10–29% are reached by a six-chain deterministic regime at matched compute, while greedy alone reaches 0–9%. The recovery scales with the deterministic budget across five graft kinds whose mechanistic distinctness we verify across all twelve cells (cross-kind fix-set Jaccard ≤ 0.47 in every cell; App. H). The decoder is unchanged: stochastic decoding and sample majority remain the right default at inference, and grafting is purely a diagnostic tool for locating the slice.

The implication is that sampling-derived difficulty annotations conflate "hard" with "unreached on the stochastic axis" on a non-trivial fraction of their hardest stratum. For AI-for-math pipelines specifically, benchmark hardness buckets, pass@k-filtered synthetic curricula, and verifier datasets, this means an identifiable fraction of items currently labeled as too hard for the generator are, at matched compute, not. The slice is structurally locatable and can be audited at a few extra deterministic forward passes per item.

Limitations

Our study covers four open-weight models (3B, 8B and 12B parameters) on three reasoning benchmarks (two free-form math, GSM8K and MATH, plus one multiple-choice, MMLU-Pro), with seven graft vectors at a single layer and position and a default sampling temperature of $T = 0.7$ (plus one $T = 0.3$ check). Even with eight deterministic chains, 66–88% of the pass@6 $= 0$ stratum remains unrecovered; those examples may be genuinely hard, or reachable along a diversity axis we do not test. We report conditional recovery as point estimates across the twelve cells, without bootstrap confidence intervals or repeated-seed-set variance; the qualitative direction holds on every cell, but per-cell rates should be read with this uncertainty in mind. We also do not study whether the recovered slice can be identified *without* gold labels at deployable precision; we treat that as a separate question. The cheap cross-chain disagreement probe in App. I works well on free-form numeric and symbolic answers (lift@5% up to $8.0\times$ on GSM8K / MATH cells) but degrades on multiple-choice benchmarks (lift@5% collapses to 1.0–$1.5\times$ across the four MMLU-Pro cells): with a bounded answer vocabulary (A–J), two sampled chains and greedy frequently agree by chance even on recoverable items, so chain disagreement loses its signal. A deployable label-free flag would need a different signal on MCQ-style outputs. Finally, on the three smallest pass@6 $= 0$ strata (`Llama-3.1-8B` / GSM8K, 51 examples, `Mistral-Nemo-12B` / GSM8K, 54, and `Qwen-2.5-3B` / GSM8K, 58), absolute recovery counts are small ($R_6 \in \{10, 11, 8\}$); the recovery *rates* line up with the other cells, but the noise floor is higher.

Acknowledgements

This work is supported by the MUR FIS2 grant n. FIS-2023-00942 "NEXUS" (cup B53C25001030001), and partly by Sapienza University of Rome via the Seed of ERC grant "MINT.AI" (cup B83C25001040001).

Impact Statement

This work is a diagnostic study of how pass@k-based hardness labels behave on math and science reasoning benchmarks; it does not propose a new inference method or a more capable model, and the activation-grafting intervention is used only as an analysis tool rather than as something to deploy. The main downstream consequence we anticipate is more careful construction of math-reasoning evaluation suites, curricula, and verifier datasets, where treating pass@$k = 0$ as intrinsic hardness can silently and erroneously drop a structurally identifiable slice of samples the model can in fact solve. Better labels of this kind should reduce wasted compute in rejection-sampling pipelines and reduce miscalibrated difficulty signals in reinforcement-learning-with-verifiable-rewards training. We do not foresee specific dual-use risks: the diagnostic does not enable new generation capabilities, does not require any training, and is run entirely on existing open-weight models and publicly released benchmarks.

References

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

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xinyun Chen, Vamsi Aribandi, Yi Tay, Dale Schuurmans, Denny Zhou, and Claire Cui. 2023. Universal self-consistency for large language model generation. In *arXiv preprint arXiv:2311.17311*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang,

Shirong Ma, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Wouter Kool, Herke Van Hoof, and Max Welling. 2019. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International conference on machine learning*, pages 3499–3508. PMLR.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*.

Noam Levi. 2026. Learning shrinks the hard tail: Training-dependent inference scaling in a solvable linear model. *Preprint*, arXiv:2601.03764.

Noam Itzhak Levi. 2025. A simple model of inference scaling laws. In *International Conference on Machine Learning*, pages 33984–33998. PMLR.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pages 39578–39601.

Llama Team. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Mistral AI Team. 2024. Mistral nemo. `https://mistral.ai/news/mistral-nemo`. 12B-parameter instruction-tuned model jointly developed with NVIDIA.

Qwen Team. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Vignav Ramesh and Kenneth Li. 2025. Communicating activations between language model agents. In *International Conference on Machine Learning*, pages 51094–51116. PMLR.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand. Association for Computational Linguistics.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *Advances in Neural Information Processing Systems*, 37:34737–34774.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Ulisse Mini, and Monte MacDiarmid. 2024. Activation addition: Steering language models without optimization.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *Transactions on Machine Learning Research*.

Table 4: Hidden-state divergence at the graft position, mean ± std over 100 GSM8K examples (`Qwen-2.5-3B`, last token, $\ell = 26$; random excludes 4 runaway-divergence outliers, $n = 96$). Layers below 26 are not shown (cos = 1, $L_2 = 0$ identically). The baseline norm at the graft position is 86.2 ± 0.9 at $\ell = 26$ and 309.2 ± 14.5 at $\ell = 35$.

	g_R (random)		g_Z (zero)	
ℓ	cos	L_2	cos	L_2
26	0.001 ± 0.021	121.8 ± 2.0	0.000	86.2 ± 0.9
27	0.130 ± 0.047	155.2 ± 8.2	0.136 ± 0.074	147.7 ± 29.9
28	0.095 ± 0.047	186.8 ± 10.8	0.071 ± 0.056	177.1 ± 23.8
30	0.003 ± 0.048	258.4 ± 16.2	0.021 ± 0.057	238.3 ± 20.3
32	-0.005 ± 0.094	341.3 ± 27.5	0.136 ± 0.165	301.2 ± 34.3
34	0.205 ± 0.130	425.7 ± 37.3	0.361 ± 0.139	370.9 ± 38.9
35	0.001 ± 0.108	466.7 ± 33.4	0.017 ± 0.073	415.1 ± 28.1

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2025. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *The Thirteenth International Conference on Learning Representations*.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

Luca Zhou, Pratham Yashwante, Marshall Fisher, Alessio Sampieri, Zihao Zhou, Fabio Galasso, and Rose Yu. 2026. Cats-bench: Can language models describe time series? *Preprint*, arXiv:2509.20823.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. 2023. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.

A Per-Layer Hidden-State Divergence (Numerical)

Table 4 reports the per-layer mean and standard deviation of cosine similarity and L_2 distance to the unperturbed run, summarised graphically in Figure 4.

B Layer-Selection Sweep

We computed the $\det^{\oplus}$ oracle ceiling on `Qwen-2.5-3B` / GSM8K at four layers $\ell \in \{2, 13, 26, 32\}$, spanning early, middle, late, and final-block positions in the 36-layer stack. The ceilings are 0.902, 0.903, 0.908, and 0.905, with corresponding $G_1 = \det^{\oplus} - g_B$ values of $+5.9\%$, $+6.0\%$, $+6.5\%$, and $+6.2\%$. G_1 remains within a 0.6% band across this 30-layer range: the diversification effect is broadly layer-insensitive, consistent with a perturbation propagating forward through the residual stream rather than acting at a privileged depth. We adopt $\ell = 26$ as the marginal best across the 3B main-text cells (sweep done on `Qwen-2.5-3B`); the same layer is used for the 8B cells, and the choice is not load-bearing.

C Pass@$k=0$ Shrink-Rate Diagnostics

A natural concern is that six sampling chains undersample the sampling regime, and that the pass@$k = 0$ stratum would collapse with more samples. Going from $k = 4$ to $k = 6$ shrinks pass@$k=0$ substantially in absolute terms, but the stratum remains 5.1–43.5% of prompts at $k = 6$, and the per-sample marginal-shrink rate is monotonically decreasing across the twelve cells (Fig. 3). Within our $k \leq 6$ budget the shrinkage has already plateaued, so the R_6 recoveries reported in Sec. 5 (10–29% of pass@6=0) are not a quirk of an undersized sampling denominator. Whether the stratum would eventually vanish at much larger k (e.g. $k = 50, 100$) is outside the budget of this study.

D Robustness to Sampling Temperature

The diagnostic holds at lower sampling temperature. On `Qwen-2.5-3B` / GSM8K with $T = 0.3$ across four seeds, the pass@$4 = 0$ stratum *grows* ($6.7\% \to 8.2\%$ of n) and the deterministic recovery on that stratum grows with it: R_6 rises from 20.9% at $T = 0.7$ to 31.7% at $T = 0.3$. Lowering T peaks the per-step distribution and reduces stochastic exploration, so the sampling-only blind spot widens while the deterministic regime continues to reach into it. The blind spot is therefore not a high-temperature artifact; if anything, the diagnostic is more visible at lower T.

E Recovery Scaling with the Deterministic Budget

Figure 5 visualises Table 3: how R_k grows from $k = 1$ to $k = 8$ across the twelve cells. Recovery

Table 5: Pass@$k = 0$ and deterministic recovery on `Qwen-2.5-3B` / GSM8K at two sampling temperatures ($n = 1000$, four matched seeds each). R_k counts examples in the pass@$4 = 0$ stratum on which at least one of k deterministic chains is correct; percentages are of the stratum. Deterministic chains are temperature-independent; the only thing that changes between columns is the sampling stratum.

	$T = 0.3$	$T = 0.7$
pass@4=0 stratum size	82 (8.2%)	67 (6.7%)
R_1 on stratum	3 (3.7%)	1 (1.5%)
R_3 on stratum	15 (18.3%)	9 (13.4%)
R_6 on stratum	26 (31.7%)	14 (20.9%)

scales near-linearly with budget on every cell, with no saturation by $k=8$.

F Per-Source Marginal Contributions

Table 6 gives the per-cell breakdown behind the claim in §5 that no single deterministic chain dominates the 3-chain oracle: for each of $\{g_B, g_Z, g_R\}$ it counts the pass@6=0 examples that chain uniquely recovers (equivalently, the drop in $\det^{\oplus}$ if it is removed).

Table 6: Per-source unique contribution to the 3-chain deterministic oracle $\det^{\oplus} = g_B \cup g_Z \cup g_R$ across all twelve cells. "Unique to x" counts examples whose gold answer is reachable through x but not the other two ($n = 1000$); equivalently, the drop in $\det^{\oplus}$ if x is removed. No single chain dominates: g_B leads on nine cells, g_R on two (both `Llama-3.1-8B`), and g_B is tied with g_Z on `Llama-3.2-3B` / GSM8K.

Setup	unique to g_B	unique to g_Z	unique to g_R
`Qwen-2.5-3B` / GSM8K	24 (2.4%)	21 (2.1%)	23 (2.3%)
`Qwen-2.5-3B` / MATH	33 (3.3%)	30 (3.0%)	21 (2.1%)
`Qwen-2.5-3B` / MMLU-Pro	68 (6.8%)	57 (5.7%)	58 (5.8%)
`Llama-3.2-3B` / GSM8K	37 (3.7%)	37 (3.7%)	31 (3.1%)
`Llama-3.2-3B` / MATH	79 (7.9%)	37 (3.7%)	53 (5.3%)
`Llama-3.2-3B` / MMLU-Pro	89 (8.9%)	61 (6.1%)	58 (5.8%)
`Llama-3.1-8B` / GSM8K	14 (1.4%)	13 (1.3%)	23 (2.3%)
`Llama-3.1-8B` / MATH	55 (5.5%)	39 (3.9%)	53 (5.3%)
`Llama-3.1-8B` / MMLU-Pro	49 (4.9%)	54 (5.4%)	57 (5.7%)
`Mistral-Nemo-12B` / GSM8K	35 (3.5%)	23 (2.3%)	21 (2.1%)
`Mistral-Nemo-12B` / MATH	71 (7.1%)	44 (4.4%)	45 (4.5%)
`Mistral-Nemo-12B` / MMLU-Pro	77 (7.7%)	52 (5.2%)	71 (7.1%)

G Examples of Recovered Items

The R_k table is a counting argument. To make the mechanism concrete, we examine three pass@6=0 examples (`Qwen-2.5-3B`) on which exactly two of $\{g_B, g_Z, g_R\}$ are correct, with the rescuing pair differing between examples. No single chain is the privileged recoverer. We characterise each case by reading the generated chains and identifying the apparent reasoning step on which the failing chain diverges from the gold-reaching one; we do not claim these descriptions are recovered from inspecting the model's internals.

GSM8K `0798` (Rescued by g_Z, g_R). *"In the next 3 years, Billy will save \$X. He's saving up for a raise from \$10/hr to \$11/hr at his current job. Meanwhile, his co-worker Sally is going to retire in 3 years at \$12/hr. If Sally earns \$1.50 more than Billy after she retires and Billy gets his raise, what's the difference between their hourly wages?"* Gold: \$20. The intended reading puts Billy at \$11/hr after his raise and Sally at \$10.50/hr after retirement (Sally earns \$1.50 more than Billy's pre-raise wage), so Billy is \$20/hr higher in scenarios that double the difference, both grafts arrive at \$20. Greedy collapses "raise" and "retire" into an averaged rate \$10.025/hr and returns \$439.50, an arithmetically clean but semantically off-axis answer; all six sampled chains return non-gold answers.

MATH Counting & Probability `0445` (Rescued by g_B, g_R). *"A mother purchases 5 blue plates, 2 red plates, 2 green plates, and 1 orange plate. How many ways are there for her to arrange these plates for dinner around her circular table if she doesn't want the 2 green plates to be adjacent?"* Gold: 588. Total circular arrangements of 10 plates with multiplicities $(5, 2, 2, 1)$ are $10!/(5!\,2!\,2! \cdot 10) = 756$; arrangements with the two greens adjacent (treat them as one block of identical greens) are $9!/(5!\,2! \cdot 9) = 168$, giving $756 - 168 = 588$. Greedy and the random graft both execute this argument cleanly. The zero graft instead treats the plates as 9 distinct objects in a circle and returns $9! - 8! \cdot 2 = 282{,}240$, dropping both the identical-colour multiplicities and the circular division. All six sampled chains return non-gold answers.

MATH Counting & Probability `0542` (Rescued by g_B, g_Z). *"Bob rolls a fair six-sided die each morning. If Bob rolls a composite number, he eats sweetened cereal. If he rolls a prime number, he eats unsweetened cereal. If he rolls a 1, then he rolls again. In a non-leap year, what is the expected number of times Bob will roll his die?"* Gold: 438. The expected rolls per morning solves $E = 1 + \frac{1}{6}E$, giving $E = 6/5$; over 365 days this is 438. Greedy and the zero graft both execute this recursion. The random graft instead sets $E = 2$ (a common slip

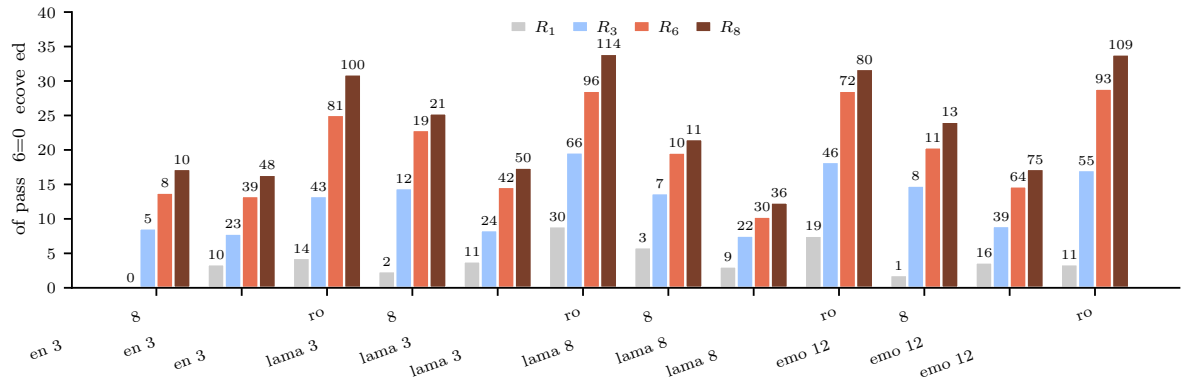


Figure 5: Deterministic recovery on the pass@6=0 slice as the deterministic budget grows from 1 to 8 chains. R_k = number of pass@6=0 examples on which at least one of k deterministic chains is correct. Numbers above bars are raw counts; percentages are of column 2 of Table 3. Recovery scales near-linearly with budget across all twelve cells.

from treating the re-roll as a single additional draw) and returns 730. All six sampled chains return non-gold answers.

Pattern. The three cases instantiate three different rescuing pairs from $\{g_B, g_Z, g_R\}$: $\{g_Z, g_R\}$, $\{g_B, g_R\}$, $\{g_B, g_Z\}$. The chain that fails is not the same chain across examples, and the chain that rescues is not the same either. The behavioural payoff of admitting multiple mechanistically distinct chains comes precisely from this lack of a privileged direction: any single chain, greedy included, misses a non-trivial slice of pass@6 = 0 that another chain in the set catches (Table 6).

H Extra Graft Vectors

As a partial empirical check on `Qwen-2.5-3B` / GSM8K, augmenting $\{g_B, g_Z, g_R\}$ with five additional grafts, mean activation, BOS-token replacement, previous-layer activation, random unit direction, and shuffled activation, all at $\ell = 26$, raises $\det^{\oplus}$ from 0.908 to 0.940, suggesting that the diversification effect is not specific to zero and random vectors.

Fix-Set Diversity. We further check that distinct grafts rescue *structurally different* examples, not the same ones in slightly different forms. For each graft g_x, define its fix-set $F_x = \{i : g_B(i) \neq \text{gold}_i,\ g_x(i) = \text{gold}_i\}$ (examples baseline gets wrong but the graft rescues). We report pairwise Jaccard over fix-sets in two views: a per-cell summary across all twelve (model, benchmark) cells (Table 7) and a full per-pair matrix on `Qwen-2.5-3B` / GSM8K as a worked example (Table 8).

Across all twelve cells, the maximum cross-kind Jaccard never exceeds 0.47, with the per-cell maximum landing in $[0.26, 0.47]$. The single same-axis pair, RANDOM vs. RANDOM_UNIT (two norm-matched Gaussian probes that by construction target the same axis), ranges from 0.21 to 0.76 across cells and exceeds the per-cell cross-kind maximum on five of twelve cells (`Qwen-2.5-3B` / GSM8K, `Qwen-2.5-3B` / MATH, `Llama-3.2-3B` / MATH, `Qwen-2.5-3B` / MMLU-Pro, `Llama-3.2-3B` / MMLU-Pro), reaching 0.76 on `Qwen-2.5-3B` / GSM8K as an illustrative ceiling for what resampling along a single random axis can contribute. On the remaining seven cells (the larger `Llama-3.1-8B` and `Mistral-Nemo-12B` rows), the same-axis pair sits within the cross-kind band rather than above it; this reflects compression of all overlaps under the bounded MCQ output space (MMLU-Pro, A–J) and the smaller per-cell fix-sets at the larger model scales, not a violation of the cross-kind ceiling, which remains ≤ 0.47 everywhere. The universal statement is the cross-kind ceiling: distinct graft kinds rescue largely disjoint slices of the pass@k = 0 stratum in every cell we test.

The `Qwen-2.5-3B` / GSM8K matrix in Table 8 illustrates the pattern at full resolution: most pairs overlap on only 28–45% of their union, including $J(F_Z, F_R) = 0.32$ between the two main-text grafts, and the single high-overlap pair ($J = 0.76$) is the same-axis RANDOM vs. RANDOM_UNIT comparison. This justifies treating g_Z and g_R as independent diversification channels rather than

Table 7: Per-cell fix-set Jaccard summary across all twelve (model, benchmark) cells. *cross-kind J*: range of pairwise Jaccard over all unordered pairs of distinct graft kinds (ZERO, RAND, RUNIT, SHUF, BOS, AVG, PREV at $\ell = 26$, last-token). *same-axis J*: Jaccard between the two random-direction probes (RAND vs. RUNIT), both norm-matched Gaussians targeting the same axis by construction. $\max |F|$: largest fix-set across the seven grafts. The `Qwen-2.5-3B` / GSM8K row corresponds to the full per-pair matrix in Table 8.

Setup	n	cross-kind J	same-axis J	$\max \lvert F \rvert$
`Qwen-2.5-3B` / GSM8K	1000	0.28–0.45	**0.76**	55
`Llama-3.2-3B` / GSM8K	1000	0.20–0.45	0.43	75
`Llama-3.1-8B` / GSM8K	1000	0.26–**0.47**	0.46	55
`Mistral-Nemo-12B` / GSM8K	1000	0.07–0.44	0.43	47
`Qwen-2.5-3B` / MATH	1000	0.21–0.43	0.46	77
`Llama-3.2-3B` / MATH	1000	0.22–0.36	0.37	91
`Llama-3.1-8B` / MATH	1000	0.22–0.42	0.31	91
`Mistral-Nemo-12B` / MATH	1000	0.08–0.27	0.24	83
`Qwen-2.5-3B` / MMLU-Pro	1000	0.19–0.29	0.45	112
`Llama-3.2-3B` / MMLU-Pro	1000	0.12–0.28	0.42	124
`Llama-3.1-8B` / MMLU-Pro	1000	0.17–0.29	0.27	97
`Mistral-Nemo-12B` / MMLU-Pro	1000	0.06–0.26	0.21	114

Table 8: Pairwise Jaccard of fix-sets on `Qwen-2.5-3B` / GSM8K, last-token, $\ell = 26$. Each cell is $|F_a \cap F_b|/|F_a \cup F_b|$, where F_x is the set of examples baseline gets wrong but graft g_x rescues. RAND=norm-matched Gaussian (=g_R), RUNIT=random unit direction, SHUF=hidden-dim permutation, BOS=BOS-token replacement, AVG=mean prompt activation, PREV=same position from layer $\ell-1$.

	ZERO	RAND	RUNIT	SHUF	BOS	AVG	PREV
ZERO	–	0.32	0.36	0.40	0.45	0.33	0.32
RAND	0.32	–	**0.76**	0.29	0.32	0.28	0.31
RUNIT	0.36	**0.76**	–	0.30	0.35	0.31	0.32
SHUF	0.40	0.29	0.30	–	0.43	0.39	0.34
BOS	0.45	0.32	0.35	0.43	–	0.40	0.44
AVG	0.33	0.28	0.31	0.39	0.40	–	0.43
PREV	0.32	0.31	0.32	0.34	0.44	0.43	–

redundant perturbations.

I Label-Free Flag: Per-Setup Numbers

Table 9 reports the per-setup precision, recall, and lift of the cheap label-free probe described in §7.2. The probe scores each item by the number of disagreeing pairs in the three-chain set $\{g_B, S_0, S_1\}$ (three forwards per item) and ranks items in descending order. Recoverable items are those in the pass@$6 = 0$ stratum on which at least one chain in $\{g_B, g_Z, g_R\} \cup \{\text{RUNIT}, \text{SHUF}, \text{BOS}\}$ is correct. Lift is precision at the operating point divided by the base rate.

J Graft Position Ablation

The main-text grafts replace the hidden state at the *last prompt token* ($p = -1$). We ablate this choice on `Qwen-2.5-3B` / GSM8K by repeating the seven-graft sweep of Appendix H at three positions: last, penultimate ($p = -2$), and first ($p = 0$); per-position oracle ceilings appear in Table 10. The BOS-token graft at $p = 0$ is a no-op (it would replace BOS by itself) and is omitted there.

The last-token position dominates by $+2.5$–3.7% of oracle ceiling and is the position at which a single graft already realises the largest gain (g_{BOS} alone reaches 0.898, i.e. $+5.5\%$). First-token grafting is competitive ($\det^{\oplus} = 0.915$) but is dominated by the AVERAGE_ACT graft alone ($+4.2\%$), suggesting that early-position perturbations act through a single mechanism rather than via several disjoint channels. Penultimate is uniformly worst. We adopt the last-token position throughout the paper on this basis.

K Matched-Cost Sample-vs-Graft Substitution

Per-cell, per-budget numbers backing the matched-cost coverage result in §7.1; Figure 7 visualises the same data, sweeping B for both the fixed-AVG policy and the oracle upper bound. At total budget B forward passes per item, the canonical sampling-only baseline ("pure S") runs the first B sample seeds (canonical order $S_{42}, \ldots, S_{50}$); the substitution ("S+D") runs the first $B-1$ samples plus the single deterministic chain that minimises residual pass@$B = 0$ count, picked from $\{g_B, g_Z, g_R, \text{RUNIT}, \text{SHUF}, \text{BOS}, \text{AVG}, \text{PREV}\}$. Lower is better; the best chain is shown in parentheses. Across the 96 (cell, B) combinations

Table 9: Cheap label-free probe (3 forwards per item) at flagging recoverable items. $n_{\mathrm{p6}}=0$: pass@$6=0$ count. n_{rec}: recoverable count. Base: n_{rec}/n. AUPRC: rank-only area under PR. P@$K\%$: precision among the top $K\%$ ranked. L@$K\%$: lift over the base rate. Top-1% cells with P=0 reflect noise: on the two smallest recoverable strata ($n_{\mathrm{rec}}=8, 10$), the single highest-scoring item happens not to be recoverable; precision stabilises by P@2%.

	Stratum sizes					Precision (%)				Lift ($\times$ base)			
Setup	n	$n_{\mathrm{p6=0}}$	n_{rec}	base	AUPRC	P@1	P@2	P@5	P@10	L@1	L@2	L@5	L@10
`Qwen-2.5-3B` / GSM8K	1000	58	8	0.8%	0.020	0.0	0.0	4.0	3.0	0.0	0.0	5.0	3.8
`Qwen-2.5-3B` / MATH	1000	293	39	3.9%	0.106	10.0	15.0	12.0	10.0	2.6	3.8	3.1	2.6
`Qwen-2.5-3B` / MMLU-Pro	1000	323	81	8.1%	0.106	30.0	20.0	12.0	9.0	3.7	2.5	1.5	1.1
`Llama-3.2-3B` / GSM8K	1000	83	19	1.9%	0.054	10.0	10.0	6.0	4.0	5.3	5.3	3.2	2.1
`Llama-3.2-3B` / MATH	1000	287	42	4.2%	0.080	10.0	15.0	12.0	6.0	2.4	3.6	2.9	1.4
`Llama-3.2-3B` / MMLU-Pro	1000	336	96	9.6%	0.140	0.0	0.0	10.0	16.0	0.0	0.0	1.0	1.7
`Llama-3.1-8B` / GSM8K	1000	51	10	1.0%	0.050	0.0	5.0	8.0	5.0	0.0	5.0	8.0	5.0
`Llama-3.1-8B` / MATH	1000	291	30	3.0%	0.083	20.0	15.0	8.0	8.0	6.7	5.0	2.7	2.7
`Llama-3.1-8B` / MMLU-Pro	1000	252	72	7.2%	0.098	0.0	10.0	10.0	11.0	0.0	1.4	1.4	1.5
`Mistral-Nemo-12B` / GSM8K	1000	54	11	1.1%	0.032	0.0	5.0	4.0	4.0	0.0	4.6	3.6	3.6
`Mistral-Nemo-12B` / MATH	1000	435	64	6.4%	0.103	0.0	10.0	10.0	10.0	0.0	1.6	1.6	1.6
`Mistral-Nemo-12B` / MMLU-Pro	1000	322	93	9.3%	0.091	10.0	10.0	10.0	8.0	1.1	1.1	1.1	0.9

Table 10: Oracle ceilings $\det^{\oplus}$ on `Qwen-2.5-3B` / GSM8K when the graft is applied at three different prompt positions; $\ell=26$ throughout. Each ceiling is the union of g_B and the per-position graft set described in the text. Last-token grafting both reaches the highest ceiling and most cleanly diversifies the ensemble.

position p	ceiling $\det^{\oplus}$	Δ over g_B	# grafts
last (-1)	**0.940**	+9.7%	7
first (0)	0.915	+7.2%	6
penultimate (-2)	0.903	+6.0%	7

summarised below, substitution wins on 80, ties on 4, and loses on 12.

The strongest substitution gains concentrate on MMLU-Pro and on the larger models within each benchmark, where the sampling axis saturates earlier (cf. Fig. 3) and the deterministic axis still contributes fresh coverage. The AVG chain is the single most-frequently-chosen substitution (32/96 slots), followed by RAND (25), SHUF (10), BOS (7), B (6), Z (6), RUNIT (6), and PREV (4); no single chain dominates, consistent with the fix-set diversity reported in App. H.

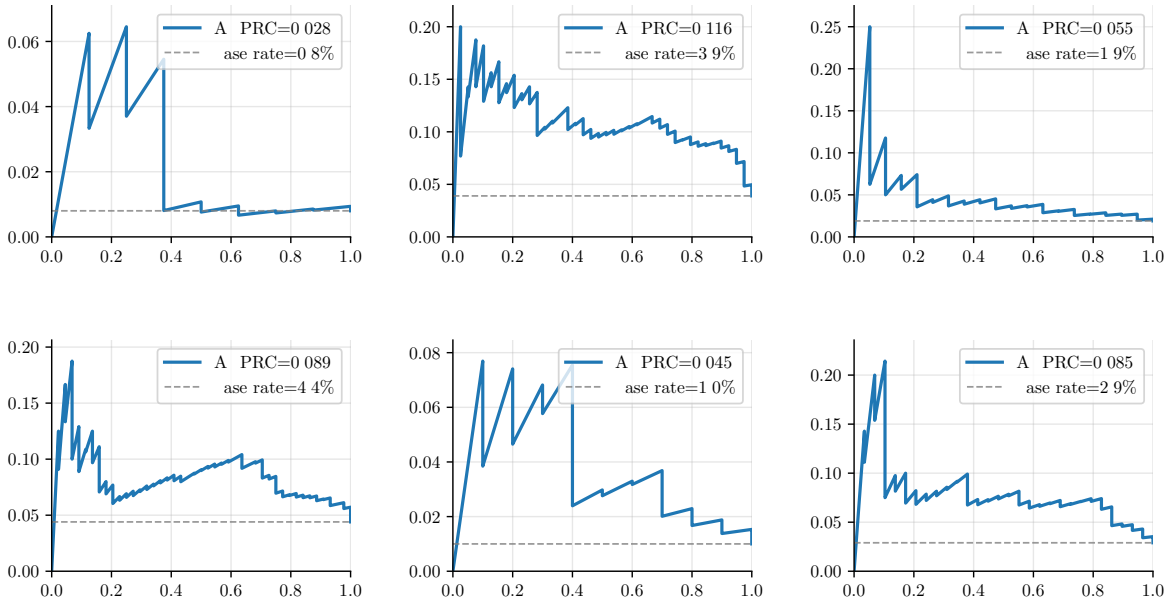


Figure 6: Precision–recall of the cheap label-free probe (g_B plus two sampling seeds, three forwards) at flagging recoverable items (pass@$6=0$ *and* at least one deterministic chain correct). Dashed line is the base rate (random selection). The probe ranks recoverable items above the base rate across all twelve cells; precisions look low absolutely because the recoverable slice itself is small (0.8–9.6% of n). Per-setup top-K% numbers and lift are in Table 9.

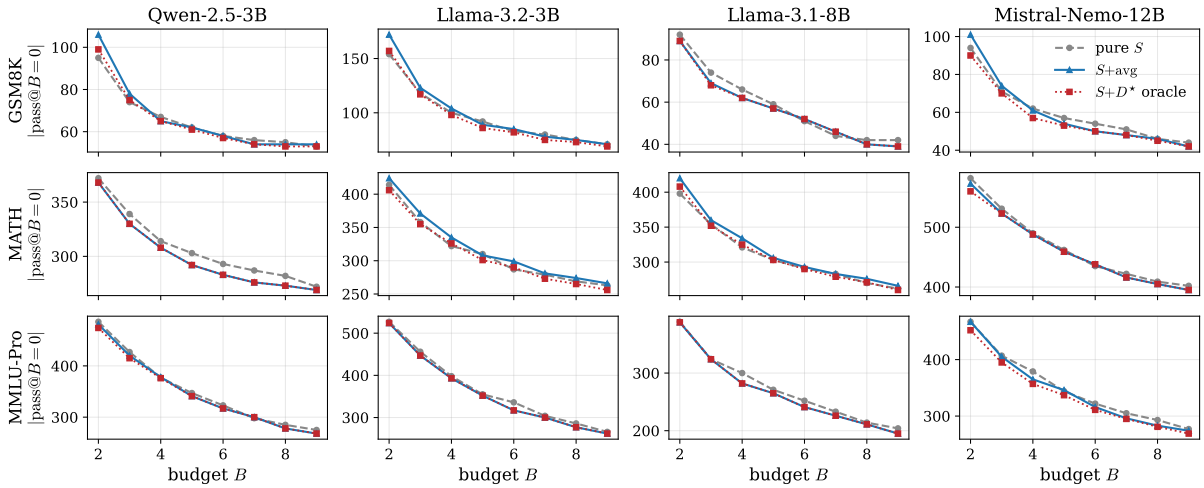


Figure 7: Matched-cost coverage: pass@$B=0$ count under pure sampling (gray dashed, "S": first B sample seeds), the proposed *fixed-policy* substitution (blue solid, "S+avg": $B-1$ samples plus the mean-prompt activation chain), and the oracle upper bound (red dotted, "$S+D^{\star}$": $B-1$ samples plus the single best deterministic chain at that (cell, B), picked from $\{g_B, g_Z, g_R, \text{RUNIT}, \text{SHUF}, \text{BOS}, \text{AVG}, \text{PREV}\}$), swept over $B \in \{2, \ldots, 9\}$. Lower is better. The fixed avg policy wins $59/96$ (cell, B) slots with 8 ties and net mean $+1.66$ items per slot; the oracle, an upper bound that picks the best chain per slot, wins $80/96$. Both lines drop below gray on every MMLU-Pro cell. Per-cell, per-B counts in Table 11.

Table 11: Pass@$B=0$ count under pure sampling ("S") vs. one-chain substitution ("S+D", best deterministic chain in parentheses) at matched compute $B \in \{2, \ldots, 9\}$. Lower is better; **bold** marks the better column at each B (ties not bolded). Per-cell unconstrained best mix $(s^\star, d^\star)$ at $B=9$ in the last column ($n = 1000$). "M.-Pro" denotes MMLU-Pro

	Setup	$B=2$		$B=3$		$B=4$		$B=5$		$B=6$		$B=7$		$B=8$		$B=9$		best mix at $B=9$
		S	S+D	S	S+D	S	S+D	S	S+D	S	S+D	S	S+D	S	S+D	S	S+D	
GSM8K	Qwen-2.5-3B	**95**	99 (BOS)	**74**	75 (PREV)	**67**	**65** (RUNIT)	62	**61** (R)	58	**57** (R)	56	**54** (AVG)	55	**53** (Z)	53	53 (Z)	$s=2, d=7$
	Llama-3.2-3B	**154**	157 (PREV)	**118**	117 (Z)	100	**98** (SHUF)	92	**86** (SHUF)	83	**82** (Z)	80	**75** (SHUF)	75	**73** (SHUF)	71	**69** (SHUF)	$s=4, d=5$
	Llama-3.1-8B	92	**89** (R)	74	**68** (B)	66	**62** (B)	59	**57** (AVG)	**51**	52 (AVG)	**44**	46 (Z)	42	**40** (AVG)	42	**39** (SHUF)	$s=7, d=2$
	Mistral-Nemo-12B	94	**90** (R)	71	**70** (R)	62	**57** (R)	57	**53** (R)	54	**50** (R)	51	**48** (SHUF)	46	**45** (SHUF)	44	**42** (SHUF)	$s=4, d=3$
MATH	Qwen-2.5-3B	372	**368** (Z)	339	**330** (AVG)	314	**308** (AVG)	303	**292** (AVG)	293	**283** (AVG)	287	**276** (AVG)	282	**273** (AVG)	272	**269** (AVG)	$s=1, d=8$
	Llama-3.2-3B	414	**406** (B)	**358**	359 (PREV)	**322**	326 (RUNIT)	310	**301** (RUNIT)	**287**	290 (PREV)	279	**273** (RUNIT)	269	**265** (RUNIT)	263	**256** (RUNIT)	$s=8, d=1$
	Llama-3.1-8B	**398**	408 (BOS)	354	**352** (R)	**321**	325 (R)	303	303 (R)	291	**290** (R)	283	**279** (R)	**270**	271 (R)	262	**260** (R)	$s=9, d=0$
	Mistral-Nemo-12B	582	**560** (B)	531	**523** (B)	490	**488** (AVG)	462	**459** (AVG)	**435**	438 (AVG)	422	**416** (SHUF)	409	**405** (R)	402	**395** (B)	$s=8, d=1$
M.-Pro	Qwen-2.5-3B	486	**474** (BOS)	427	**415** (BOS)	377	**376** (BOS)	347	**341** (AVG)	323	**317** (AVG)	**298**	300 (AVG)	285	**278** (AVG)	275	**268** (AVG)	$s=7, d=2$
	Llama-3.2-3B	527	**524** (BOS)	456	**447** (AVG)	398	**393** (AVG)	355	**352** (AVG)	336	**317** (AVG)	304	**300** (AVG)	286	**277** (AVG)	266	**262** (AVG)	$s=7, d=2$
	Llama-3.1-8B	389	**388** (AVG)	324	324 (AVG)	300	**282** (R)	271	**265** (R)	252	**241** (AVG)	233	**226** (AVG)	214	**211** (AVG)	204	**195** (AVG)	$s=9, d=0$
	Mistral-Nemo-12B	467	**452** (BOS)	407	**395** (R)	379	**357** (R)	344	**337** (R)	322	**311** (R)	305	**295** (R)	293	**281** (R)	277	**269** (R)	$s=2, d=7$

BlueprintRepair: Typed Local Edits for Failed Lean Proof Blueprints

Ruslan Khrulev
Lomonosov Moscow State University
Moscow, Russia
ra.khrulev@gmail.com

Abstract

LLM-based Lean proving systems increasingly organize a proof as a *blueprint*: a dependency graph of formal statements. We introduce BLUEPRINTREPAIR, a repair interface that lets a model change this graph through ten schema-checked local operations. An operation names the node it edits, so the target theorem cannot be changed. Lean checks every applied change, and an accepted repair must declare every blueprint lemma its proof uses. We also release BLUEPRINTTRACE, a benchmark of 142 controlled failures with complete accepted and rejected repair trajectories. We compare typed edits, exact source patches, and complete module rewrites under matched source, feedback, model, and budget, one episode per state and interface. With DeepSeek-V4-Flash, the three interfaces solve almost the same number of the benchmark's localized failures. Typed repair is the cheapest per solved state (patching is $1.30\times$ as expensive, rewriting $2.06\times$), and within 10,000 completion tokens per task it reaches almost all of its final coverage, while both free-form interfaces are well behind. A second model, Qwen3.6-Flash, solves fewer states but keeps typed repair cheapest, puts it ahead on the proof-authoring states, and repeats the localized pattern. Code, benchmark, and trajectories are available at https://github.com/Karifannaa/BlueprintRepair.

1 Introduction

LLM-based theorem provers have traditionally generated tactics or complete proofs (Yang et al., 2023; First et al., 2023; Ren et al., 2025). Newer systems first build a structured proof plan. In Lean (de Moura and Ullrich, 2021), this plan can be represented as a *blueprint*: a dependency graph whose nodes are formal statements and whose edges record which statements a proof is meant to use. LeanArchitect maintains this metadata inside Lean developments (Zhu et al., 2026); Goedel-Architect and LeanMarathon use evolving blueprints to coordinate longer proving and formalization processes (Chung et al., 2026; Zhang et al., 2026b).

Once the proof plan is explicit, repair need not begin by regenerating the whole Lean file. A failed blueprint may contain a false intermediate lemma, a missing dependency, or an unused node. In these cases, a small change to the existing graph may be enough. A different failure occurs when all statements and edges are correct, but one lemma still lacks a proof. We ask: **when are typed local edits sufficient, and when is freer Lean code generation more useful?**

Revising a failed proof plan is not new (Chung et al., 2026; Xiao et al., 2026); what differs is the editable object: a declaration-level LeanArchitect graph changed through schema-checked operations rather than freshly written Lean text. We propose BLUEPRINTREPAIR, a typed local-edit interface for failed Lean blueprints. At each step, the model chooses one of ten operations that can change a statement, add or remove an edge, split or delete a node, or write a proof for one node. The harness applies the operation mechanically and returns Lean feedback. We compare this interface with two free-form alternatives: atomic search-and-replace patches over the current source and complete module rewriting. All three see the same normalized source, typed state, verifier feedback, model, and budget. The only intended difference is how a repair is expressed.

A kernel-accepted target theorem is necessary but not sufficient for blueprint repair. Suppose the proof of a node uses another blueprint lemma but the graph does not declare that dependency. Lean may still accept the theorem, while the blueprint remains wrong. We therefore inspect each compiled proof term and reject any repair with an undeclared blueprint dependency. This check directly addresses the central distinction of the paper: we evaluate repaired proof graphs, not only repaired

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 30–41
October 29, 2026 ©2026 Association for Computational Linguistics

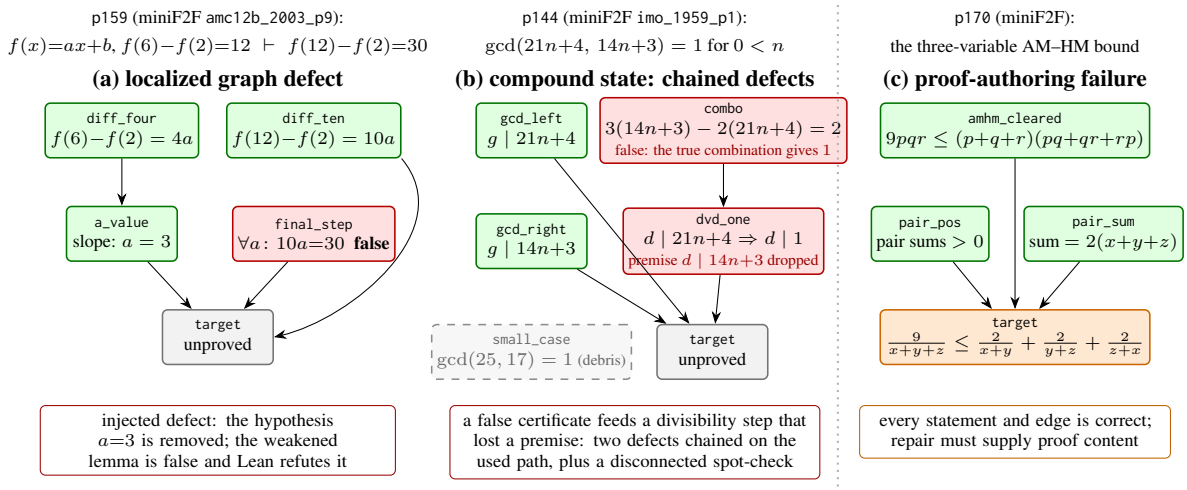


Figure 1: Three benchmark examples. (a) A localized defect weakens a lemma until it becomes false. (b) A compound state contains two linked defects on the target path and one disconnected node. (c) In a proof-authoring state, the graph is correct but the remaining theorem still needs proof content. Statements are abbreviated; Table 1 lists all failure families.

target proofs.

On the 91 states with localized graph or statement defects, typed repair solves 79 states with DeepSeek-V4-Flash (DeepSeek-AI, 2026a,b) and each free-form interface solves 81; a second model, Qwen3.6-Flash (Qwen Team, 2026), shows the same pattern. The small coverage gap comes with a large efficiency difference. Typed repair reaches 103 total solves within 10,000 completion tokens per task, compared with 90 for patching and 87 for rewriting, and has the lowest provider cost per solve. Patching obtains the highest observed total, though no pairwise difference is clear of zero, and each interface is the sole solver of a few states, so the three together cover more than any one of them. These results support a simple systems view: when a blueprint is mostly correct, typed local edits recover most of the reachable coverage at the lowest cost, and free-form generation buys the remainder at a higher price.

Our main contributions are:

- **A typed local-repair interface.** Ten schema-checked operations edit statements, nodes, edges, and proofs while Lean verifies every resulting state.
- **A graph-aware acceptance criterion.** In addition to preserving and proving the target, an accepted module must declare every inter-node dependency found in its compiled proof terms.
- **A controlled three-interface study.** Typed edits, local source patches, and whole-module rewrites receive equal source access and feedback, allowing a direct comparison of coverage, token use, cost, and control.
- **A trajectory release.** BLUEPRINTTRACE contains the failed states, construction metadata, accepted and rejected actions, Lean feedback, graph checks, token and cost records, and final modules.

2 Repair task and benchmark

What counts as a repaired blueprint. A blueprint is a directed acyclic graph over a Lean module. An edge $u \to v$ means that the proof of node v declares node u as a dependency. Let C be the target theorem together with every node that can reach it along declared edges. A state is accepted only when (i) the full module elaborates, (ii) every node in C is proved, (iii) the target statement is unchanged and contains no forbidden proof shortcut, (iv) no node is a stored kernel-refuted statement, and (v) the compiled proofs agree with the declared graph.

The last condition is easiest to state for one proved node v. $\mathrm{Declared}(v)$ is the set of blueprint nodes connected to v by its incoming `proofUses` edges. $\mathrm{Actual}(v)$ is the set of blueprint nodes reached from the elaborated proof term of v; the extractor follows ordinary helper definitions and stops when it reaches another blueprint node. We require

$$\mathrm{Actual}(v) \subseteq \mathrm{Declared}(v).$$

Table 1: Controlled failure families. Excluding the 12 compound states, the first five families hold the 91 edit-shaped states and the last two the 39 proof-authoring states. A compound state is counted under the family of its first failing defect but evaluated as a separate stratum. Example repairs are possibilities, not required routes.

Family	n	Injected defect	Example typed repair
false / too strong	23	false statement on the target path	weaken, rewrite, or drop it
missing hypothesis	34	required assumption removed	strengthen hypotheses
missing dependency	18	proof uses an undeclared lemma	add the missing edge
redundant dependency	14	unnecessary node wired into the path	remove edge or node
dead node	14	node lies outside the target closure	remove the unused node
monolithic	31	intact node not closed by the prover	split or prove the node
representation	8	inconvenient formal representation	restate and prove

Table 2: Graph-size ranges in the controlled benchmark.

Stratum	n	nodes	edges	max depth
all	142	1–8	0–8	3
edit-shaped	91	2–7	0–6	3
proof-authoring	39	1–5	0–4	1
compound	12	5–8	3–8	3

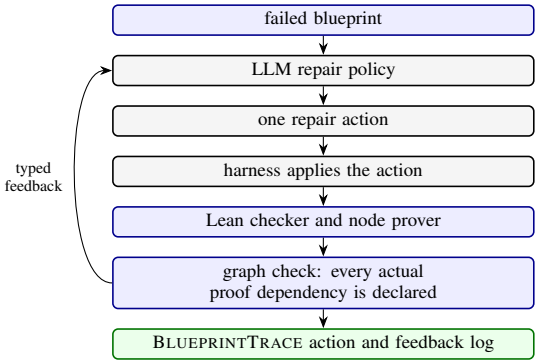


Figure 2: The repair loop used by all three interfaces. The harness applies one candidate action, Lean checks the resulting module, and the graph check returns structured feedback for the next step.

Thus a proof may not use an undeclared blueprint lemma. We do not require exact equality because a repaired proof can stop using an edge that is still harmlessly declared; we record such unused edges as a separate quality signal. Nodes outside C must still elaborate and must not be refuted, but they do not contribute to the target proof and may remain deferred. We separately report whether every node in the module is complete. Appendix B states the check in full.

Two repair regimes. Figure 1 shows the distinction measured by the benchmark. In an *edit-shaped* failure, at least one statement, node, or edge is wrong. A proof-only change may bypass the defect, but the failed graph itself contains a known structural error. In a *proof-authoring* failure, the statements and edges are intact, but the fixed node prover does not close one of the required nodes. The label is operational: it describes failure under the fixed prover used by the harness. Appendix B.2 reports a stronger-automation baseline on exactly these states.

Controlled states. The benchmark contains 142 failed blueprints over 141 distinct miniF2F target theorems (Zheng et al., 2022). The targets come from the miniF2F Test and Valid splits at a pinned commit (244 items each). A mechanical filter on the statement's type surface removes 86 analysis-style items; from the remaining pool of 402 we chose targets manually, balancing failure families and topic areas, and one target contributes two independently constructed states. We start from correct, author-constructed blueprints and introduce defects from Table 1. Each state is checked before any model run: false statements have Lean refutations, target statements are preserved, dependency defects are confirmed against compiled proof terms, monolithic states keep their original statements and edges, and representation states swap one statement for a true but inconvenient form. The model is not required to undo the construction edit; any final module is valid when it proves the original target and passes the acceptance criterion above.

Of the 142 states, 91 contain one localized edit-shaped failure, 39 are proof-authoring failures, and 12 are compound states with linked defects. The graphs are intentionally small so that the experiment isolates repair rather than long-horizon search; Table 2 gives their size ranges.

3 The BLUEPRINTREPAIR interface

Figure 2 shows the complete interaction loop. The model receives the current blueprint, the current Lean source, and typed diagnostics. It emits one action in the format required by its interface. The harness applies the action, elaborates the module, runs the fixed node prover on deferred nodes, checks actual proof dependencies against declared edges, and returns a short reason for every accepted or rejected step. The new state then becomes the input to the next turn. All transitions are stored in

Table 3: The ten typed operations, grouped by purpose.

Purpose	Operations
change a statement	`weaken_node`, `strengthen_hypotheses`, `rewrite_node_statement`
change graph structure	`split_node`, `add_dependency`, `drop_dependency`
remove a node	`drop_false_node`, `drop_dead_node`
write proof content	`set_node_proof`
end the episode	`stop_unrepairable`

Table 4: Interface properties. All three interfaces see the same current source, typed diagnostics, and verifier feedback.

Property	Typed	Patch	Rewrite
output unit	one operation	edit blocks	full module
untouched source preserved	mechanically	mechanically	no
semantic action schema	yes	no	no
target guard fires	before applying	after applying	after applying

BLUEPRINTTRACE.

Typed local edits. Our proposed interface exposes the ten operations in Table 3. A call must satisfy a JSON schema and refer to existing nodes. The harness rejects unknown names, cycles, forbidden constructs, target statement changes, and inconsistent proof dependencies. The operations are local: source outside the requested edit is preserved mechanically.

Two free-form comparisons. The local-patch interface returns one or more exact search-and-replace blocks. All blocks are applied atomically, so a failed match leaves the module unchanged. The rewrite interface returns a complete replacement module. Patch and rewrite candidates pass the same parser, allowed-import, namespace, forbidden-token, target-signature, Lean, and graph-dependency checks as typed repairs. Table 4 summarizes what remains different. In particular, patching controls for the main concern with a rewrite-only baseline: it sees the full source and can make a small free-form edit without regenerating untouched code.

Matched evaluation protocol. The main experiment gives every interface the same normalized module, typed state, verifier feedback, pinned DeepSeek-V4-Flash model, output cap, and interaction budget. Each step may use one model response; the limit is eight steps for ordinary states and twelve for compound states. A candidate counts as solved only after the online acceptance checks described in Section 2. Appendix C gives the configuration and the three system prompts in full; the schemas, Lean versions, token accounting, and pricing snapshots are released with the artifact.

Table 5: Solved controlled states. A state counts only if the repair is accepted during the run and the stored file passes the same later dependency check. Both models see the same states and are counted the same way.

State type	n	Typed	Patch	Rewrite
DeepSeek-V4-Flash				
edit-shaped	91	79	81	81
proof-authoring	39	16	18	13
compound	12	9	10	10
total	142	104	109	104
Qwen3.6-Flash				
edit-shaped	91	70	71	72
proof-authoring	39	10	7	7
compound	12	4	10	5
total	142	84	88	84

A second model. We repeat the same evaluation with Qwen3.6-Flash. The states, prompts, interfaces, budgets, output cap, and acceptance checks are unchanged.

4 Results

4.1 Coverage

Table 5 reports the controlled benchmark. A state counts as solved only if the repair is accepted during the run and the stored file passes the dependency check we repeat afterwards (end of this subsection). With DeepSeek-V4-Flash, typed local repair solves 79 of the 91 edit-shaped failures, only two fewer than either free-form interface. Patching and rewriting solve 10 of the 12 compound states, and typed repair solves 9. On the 39 proof-authoring failures, local patching leads with 18, typed repair solves 16, and rewriting solves 13. Across all 142 states, patching has the highest final coverage (109), followed by typed repair and rewriting (104 each).

These gaps are small relative to the sample. On the primary endpoint, the 91 edit-shaped states, typed repair and patching disagree on four states only: a paired difference of -2.2 points with a 95% interval of $[-6.6, +2.2]$ (exact McNemar $p = 0.63$, source theorems resampled), and no pairwise difference on any stratum is clear of zero. The interval covers uncertainty over states of this construction, not over repeated samples from the model, and an interval covering zero is not evidence of equivalence.

The overlap is also informative. Typed repair and patching solve 78 edit-shaped states in common;

one is solved only by typed repair and three only by patching. Typed repair and rewriting solve 77 in common; two are solved only by typed repair and four only by rewriting. Thus the methods are not only different encodings of the same successes: each free-form interface adds a small number of localized repairs, while typed repair also closes cases that each of them misses.

Over all 142 states, 89 are solved by all three interfaces and 24 by none; Table 7 in Appendix B.3 splits both by injected family. Each interface is the sole solver of a few: two states are solved only by typed repair, three only by patching and three only by rewriting. The three interfaces together cover 118 of the 142 states.

Repeating the dependency check afterwards. During a run, a step is refused if the proof it produces uses a blueprint lemma for which no edge is declared. That check sees the module the harness builds. We therefore repeat it on the files we store: every stored repair that can be elaborated again gets one more Lean run, 317 in all, comparing the lemmas each proof actually uses against the edges the blueprint declares. No stored repair hides a dependency: 104 of 104 typed, 109 of 109 patch, and 104 of 104 rewrite results pass. The reverse is not true: some repairs keep declared edges that their proofs no longer use; Appendix B reports them.

The second model. Qwen3.6-Flash solves fewer states in every interface. The comparison between interfaces still repeats. On the 91 edit-shaped failures the three interfaces stay within two states of each other (70 typed, 71 patch, 72 rewrite). Two results do not repeat. On the 12 compound states typed repair solves 4 while patching solves 10, and on the 39 proof-authoring states typed repair leads with 10 against 7 for both free-form interfaces. The later dependency check is again passed by every stored repair: 84 of 84 typed, 88 of 88 patch, and 84 of 84 rewrite.

4.2 Efficiency under equal budgets

Panels (a) and (b) of Figure 3 report the matched DeepSeek-V4-Flash run. The typed interface reaches useful coverage earlier: with at most 10,000 completion tokens per task it has already solved 103 states, compared with 90 for patching and 87 for rewriting. This is almost the full typed total of 104. Patching eventually reaches 109 and overtakes the typed curve only after 34,767 completion tokens per task. Rewriting reaches the same total of 104 as typed repair, but needs 96,944 completion tokens per task to get there.

The same pattern appears in provider cost. Typed repair has the lowest all-in cost per accepted solve: relative to it, patching is $1.30\times$ as expensive and full rewriting is $2.06\times$ as expensive. The comparison includes unsuccessful episodes and all input and output charges. The result is not merely that typed responses are shorter: the frontier asks how many states are solved under the same cumulative budget and therefore combines response length, number of attempts, failures, and early stopping.

The second model. Panels (c) and (d) repeat the reading for Qwen3.6-Flash. The ordering between interfaces is the same and the gaps are wider: per accepted solve, patching costs $2.32\times$ and rewriting $3.17\times$ as much as typed repair ($0.027 per solve for typed repair, $0.062 and $0.085 for the free-form interfaces).

4.3 Control and repair behavior

All three interfaces keep the target statement, but they stop a change at different points. A typed operation names the node it edits, so an attempt on the target is refused before anything is applied; three such calls occur in the matched run and all are refused. Patching and rewriting write Lean text and are checked afterwards against the target signature: rewriting returns a module in which no node carries the target statement in 12 responses, all rejected, and every patch keeps the target statement in this run. The graph-dependency gate also rejects two typed, five patch, and one rewrite candidates that would otherwise leave proof use inconsistent with the declared blueprint; as reported above, none of the accepted results violates that rule either.

The interfaces preserve source at different granularities. A typed operation changes only the requested graph object, and a patch changes only text matched by its blocks. Rewriting may replace the whole decomposition. We treat graph size, disconnected nodes, unused edges, statement changes, and full-module completion as separate quality dimensions rather than collapsing them into the solve label; Appendix B.4 reports what each interface leaves behind.

The remaining gap is not a missing operation. Patching solves 11 states that typed repair does not. Either the typed belt lacks an operation those repairs need, or it has the operations and the model

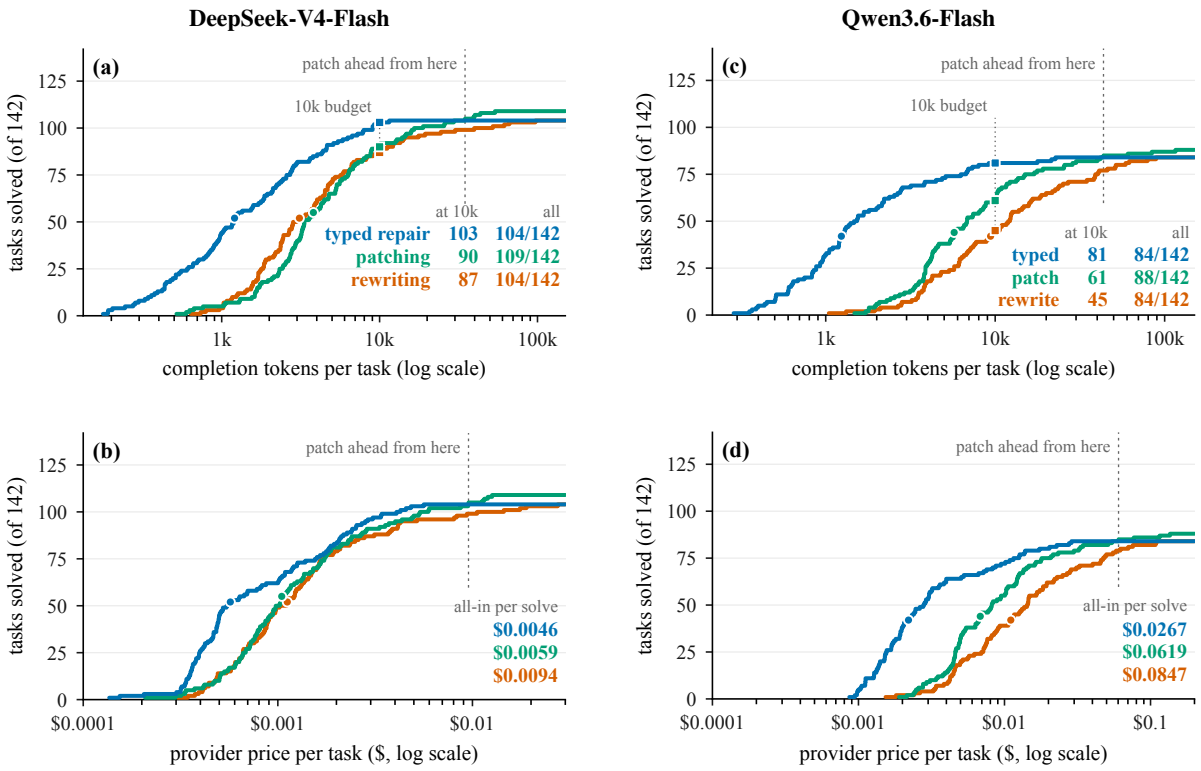


Figure 3: Cumulative solved states as the per-task budget increases, for both models. Panels (a) and (c) use completion tokens; panels (b) and (d) use provider price. All four use the same 142 initial states, count solved as in Table 5, and stop charging a task when it is first solved. The token axes are identical, so the two models' left-hand panels can be read against each other; the price axes are not, because Qwen3.6-Flash output is priced an order of magnitude above DeepSeek-V4-Flash's.

chose badly. To tell these apart we take each accepted patch, write by hand the typed calls that produce the same repair, and run them through the same harness and the same checks; no model is involved. All 11 repairs can be expressed: 10 reproduce the final module exactly, and the eleventh differs only in declaration names, which no typed operation changes. The longest takes 6 calls, every program fits the budget its own episode had, and every intermediate state passes the checks. The gap is therefore in which actions were chosen, not in which actions exist. This shows that the repairs are reachable, not that the model would have found them. Both interfaces saw the same source, and no program reuses a proof body the typed side was not shown.

Combining the interfaces. Replaying the runs we already have in a fixed order (each interface starting again from the initial failed state, a task stopping at its first accepted repair) reaches 118 of the 142 states. Every order reaches that number, but starting with the typed interface is the cheapest ($1.41 against $1.66 for the most expensive order).

5 BLUEPRINTTRACE: states, trajectories, and audits

BLUEPRINTTRACE is the data counterpart of the repair harness. The release contains all 142 controlled states as a single table: the source theorem, the initial Lean module, the graph, typed node statuses, and construction metadata. The metadata records the correct source blueprint or the documented injected defects and the Lean checks that support the assigned family. This information explains how the state was created; it does not prescribe the repair that a model must produce.

For each interface, an episode stores every model response, whether it was applied, the exact graph and source change, Lean and node-prover feedback, the actual-versus-declared dependency scan, token and cost usage, and the final module. Rejected actions are retained with their reasons. These rows show which repair decisions were attempted but invalid in a particular state: for example, an edge that would create a cycle, a proof that uses an undeclared node, or a rewrite that changes the target. A release containing only successful final proofs

Table 6: BLUEPRINTTRACE at a glance.

controlled states	142 failed blueprints over 141 miniF2F targets; seven families plus compound states
per-step records	source, graph, action, applied change, Lean feedback, dependency scan, tokens, and cost
terminal records	final module, target certificate, strict completion, graph quality, and solve label
reproducibility	exact prompts, all ten schemas, model identifiers, the pinned Lean environment, and analysis scripts

could not recover these decisions later.

The artifact also contains the exact system prompts, the machine-readable JSON schemas for all ten typed operations, the pinned Lean environment, the model identifiers, pricing snapshots, and scripts that regenerate the reported tables and frontier curves. This makes the interface contract and the evaluation predicate inspectable, not just the final solve labels. [1]

6 Related work

Editable plans and blueprints. EditableSketch and SketchRefine preserve proved subgoals while correcting or further decomposing a proof sketch (Xiao et al., 2026). This is the closest motivation to our local-repair view. The editable object differs: BLUEPRINTREPAIR operates on a LeanArchitect declaration-level DAG and can change formal statements, proofs, nodes, and explicit dependency metadata. Our additional focus is a schema-checked action interface, controlled graph-defect families, an actual-versus-declared proof invariant, and trajectories that retain rejected edits. Goedel-Architect, LeanMarathon, LEAP, and difficulty-aware decomposition also revise proof plans with verifier feedback (Chung et al., 2026; Zhang et al., 2026b; Kung et al., 2026; Zhang et al., 2026a).

Proof repair and verifier feedback. Baldur and APOLLO repair proof text from previous attempts and compiler messages (First et al., 2023; Ospanov et al., 2025); APRIL and OProver turn failed attempts into supervision or iterative context (Wang et al., 2026; Ma et al., 2026). Proof Repair across Type Equivalences transforms Coq proof terms after type changes (Ringer et al., 2021); our task instead repairs a failed Lean blueprint and its dependency metadata. VERITAS and process-verified reinforcement learning use Lean feedback for search or credit assignment (Acharya et al., 2026; Kim and Yun, 2026).

Process data, cost, and benchmark checks. FormalRewardBench injects controlled proof errors to evaluate reward models (Uluşan et al., 2026); BLUEPRINTTRACE adds stateful graph actions and typed rejection reasons. Cost-aware agent control chooses whether to continue or restart a proof plan (Rögnvaldsson et al., 2026); our frontier instead compares three repair interfaces under matched budgets. Recent benchmark audits show why a kernel-correct target alone is not a complete evaluation guarantee (Ammanamanchi et al., 2026). This motivates our preserved target provenance, explicit graph-dependency check, and released audit records.

7 Conclusion

Agents already repair code through a loop of tool calls. BLUEPRINTREPAIR asks what changes when the tools are ten typed operations on a proof graph rather than free-form edits to a file, and when a repair is accepted only if its compiled proof agrees with the declared graph. With DeepSeek-V4-Flash, on 91 localized failures, typed edits solve 79 states versus 81 for both local patching and complete rewriting. Across all 142 states, patching reaches the highest observed total (no pairwise coverage difference is clear of zero), but typed repair reaches almost all of its coverage much earlier: 103 solves at a 10,000-completion-token budget, compared with 90 and 87. It also has the lowest cost per solve; patching is $1.30\times$ as expensive and rewriting $2.06\times$ as expensive. A second model, Qwen3.6-Flash, solves fewer states and keeps the same cost ordering, although its coverage pattern breaks on the states that carry more than one defect. Finally, typed repair refuses a target statement change before it is applied, and every accepted endpoint is checked for undeclared proof dependencies. BLUEPRINTTRACE exposes the full interaction data needed to learn better repair policies from both successful and rejected structural edits.

Limitations

Small controlled graphs. The benchmark isolates local repair on blueprints with one to eight nodes and target paths of depth at most three. The compound set contains only twelve states. Results may change on research-scale formalizations with

[1] Code and trajectories: https://github.com/Karifannaa/BlueprintRepair. Benchmark: https://huggingface.co/datasets/Karifannaa/BlueprintTrace.

longer paths, more shared definitions, and defects introduced over many refinement rounds.

Stochastic model behavior. The reported table uses one episode per state and interface, so it describes the observed runs rather than an expectation over repeated samples. Repeating the same three-way protocol with a second model tests whether the qualitative pattern transfers across model families, but it does not replace multi-seed estimation within one model.

Operational proof-authoring label. A state is called proof-authoring when the fixed `rfl`/`simp`/`omega` node prover does not close an otherwise intact graph: no statement is false and no dependency is wrong. The label describes the harness and budget, not intrinsic mathematical difficulty. Stronger automation does not remove the regime: five standard tactics close 3 of the 39 states, and only one of those is a state that no interface solved.

Implemented interfaces are bundles. Typed repair, local patching, and rewriting differ in several linked ways: action schema, output granularity, preservation of untouched source, and how target immutability is enforced. A patch reply may contain several search-and-replace blocks; a typed turn applies exactly one operation. Equal step budgets therefore do not mean equal numbers of small edits; the token and cost comparisons include this difference and do not isolate it. The experiment compares these complete interfaces. It does not isolate a causal effect of typing alone, although the local-patch arm separates locality from complete regeneration more directly than a rewrite-only comparison.

Tool vocabulary and training. The ten operations are one practical vocabulary, not a proof that no better granularity exists. The evaluated models were not trained to use it. Several mathematical repairs can be expressed by multiple action sequences, so future training should allow multiple valid routes and use rejected-action reasons as additional supervision.

Benchmark provenance. Targets come from miniF2F and may have appeared in model training data. Correct blueprints and injected defects are author-constructed and reviewed with mechanical Lean checks, but there is no second independent annotator. BLUEPRINTTRACE is a controlled diagnostic benchmark, not a sample of failures from a real generation pipeline. The results say where typed edits work well (on mostly-correct blueprints with localized defects), not that typed edits win on arbitrary blueprint failures.

References

Manish Acharya, Zhenyu Liao, Yueke Zhang, Kevin Leach, Yu Huang, and Yifan Zhang. 2026. VERITAS: Verifier-guided proof search for zero-shot formal theorem proving. *arXiv preprint arXiv:2606.19399*.

Pawan Sasanka Ammanamanchi, Siddharth Bhat, and Stella Biderman. 2026. Faults in our formal benchmarking: Dataset defects and evaluation failures in Lean theorem proving. In *Proceedings of the 43rd International Conference on Machine Learning*. https://arxiv.org/abs/2606.29493.

Jui-Hui Chung, Ziyang Cai, Zihao Li, Qishuo Yin, Rohit Agarwal, Simon Park, Rodrigo Porto, Narutatsu Ri, Ziran Yang, Shange Tang, et al. 2026. Goedel-Architect: Streamlining formal theorem proving with blueprint generation and refinement. *arXiv preprint arXiv:2606.06468*.

Leonardo de Moura and Sebastian Ullrich. 2021. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, volume 12699 of *Lecture Notes in Computer Science*, pages 625–635. Springer. https://doi.org/10.1007/978-3-030-79876-5_37.

DeepSeek-AI. 2026a. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*.

DeepSeek-AI. 2026b. DeepSeek-V4-Flash model card. https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash.

Emily First, Markus N. Rabe, Talia Ringer, and Yuriy Brun. 2023. Baldur: Whole-proof generation and repair with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1229–1241.

Minsu Kim and Se-Young Yun. 2026. Process-verified reinforcement learning for theorem proving via Lean. *arXiv preprint arXiv:2606.20068*.

Po-Nien Kung, Linfeng Song, Dawsen Hwang, Jinsung Yoon, Chun-Liang Li, Simone Severini, Mirek Olšák, Edward Lockhart, Quoc V. Le, Burak Gokturk, et al. 2026. LEAP: Supercharging LLMs for formal mathematics with agentic frameworks. *arXiv preprint arXiv:2606.03303*.

David Ma, Kaijing Ma, Shawn Guo, Yunfeng Shi, Enduo Zhao, Jiajun Shi, Zhaoxiang Zhang, Gavin Cheung, Jiaheng Liu, and Zili Wang. 2026. OProver: A unified framework for agentic formal theorem proving. *arXiv preprint arXiv:2605.17283*.

Azim Ospanov, Farzan Farnia, and Roozbeh Yousefzadeh. 2025. APOLLO: Automated LLM and Lean collaboration for advanced formal reasoning. *arXiv preprint arXiv:2505.05758*.

Qwen Team. 2026. Qwen3.6-Flash model card. https://www.qwencloud.com/models/qwen3.6-flash.

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, et al. 2025. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*.

Talia Ringer, RanDair Porter, Nathaniel Yazdani, John Leo, and Dan Grossman. 2021. Proof repair across type equivalences. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pages 112–127.

Kári Rögnvaldsson, Chenhao Sun, Jasper Dekoninck, and Martin Vechev. 2026. Optimizing the cost-quality tradeoff of agentic theorem provers in Lean. *arXiv preprint arXiv:2606.04883*.

Zeynel A. Uluşan, Burak S. Akbudak, Can S. Erer, and Gözde Gül Şahin. 2026. FormalRewardBench: A benchmark for formal theorem proving reward models. *arXiv preprint arXiv:2605.10141*.

Evan Wang, Simon Chess, Daniel Lee, Siyuan Ge, Ajit Mallavarapu, Jarod Alper, and Vasily Ilin. 2026. Learning to repair Lean proofs from compiler feedback. *arXiv preprint arXiv:2602.02990*.

Zikai Xiao, Hanzheng Wang, Meng-Hao Guo, Shi-min Hu, and Shing-Tung Yau. 2026. Editable proof sketch for automated theorem proving. In *Proceedings of the 43rd International Conference on Machine Learning*.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*.

Ning Zhang, Nongyu Di, Zenan Li, Yuan Yao, and Xiaoxing Ma. 2026a. Planning to hammer: Difficulty-aware decomposition for automating Rocq proofs. *arXiv preprint arXiv:2606.17981*.

Yuanhe Zhang, Yuekai Sun, Taiji Suzuki, Jason D. Lee, and Fanghui Liu. 2026b. LeanMarathon: Toward reliable AI co-mathematicians through long-horizon Lean autoformalization. *arXiv preprint arXiv:2606.05400*.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. miniF2F: A cross-system benchmark for formal Olympiad-level mathematics. In *International Conference on Learning Representations*.

Thomas Zhu, Pietro Monticone, Jeremy Avigad, and Sean Welleck. 2026. LeanArchitect: Automating blueprint generation for humans and AI. *arXiv preprint arXiv:2601.22554*.

Appendices

A A representative split-node repair

A.1 Splitting a hard induction node

State `p40` asks to prove that, for every natural number n,

$$12 \mid 4^{n+1} + 20.$$

The failed blueprint contains this fact as one monolithic, unproved target node. The fixed node prover does not close it, although the statement is correct. The model therefore calls `split_node` and turns the single node into a small induction blueprint.

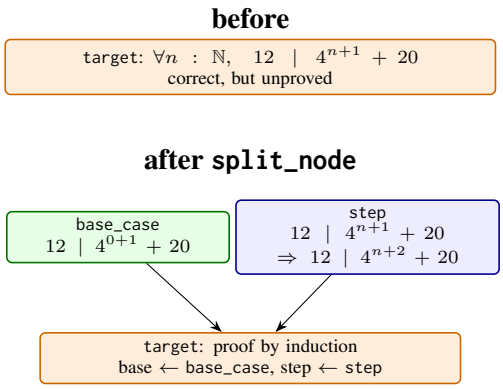


Figure 4: The `p40` repair. A monolithic induction goal is replaced by a base case and an induction step.

The split is structurally correct, but the induction step still needs a proof. The complete five-call trajectory is:

Step	Operation	Lean feedback / result
1	`split_node`	refused before anything is applied: the proposed proof uses `native_decide`
2	`split_node`	applied; `base_case` closes at once, and the proof offered for `step` does not elaborate
3	`rewrite_node_statement` on `step`	applied; a second proof for the same statement, still not elaborating
4	`set_node_proof` on `step`	`step` proved by `omega`
5	`set_node_proof` on the target	the target closes by induction; the state is solved

This example shows that typed repair is not limited to deleting or rewiring nodes: `split_node` can introduce a useful proof decomposition, after which local proof edits complete it. The refused first call is ordinary rather than exceptional: the screen that protects the target statement also rejects forbidden proof constructs before a candidate reaches Lean.

B Acceptance and benchmark details

B.1 Graph-aware terminal acceptance

For the target closure C defined in Section 2, the harness accepts an assembled module only if all of the following checks pass:

1. The complete module elaborates in a fresh Lean process without an error or timeout.
2. No blueprint node matches a stored kernel refutation certificate.
3. Every node in C is proved; proofs produced by the fixed node prover are spliced into the module and the assembled module is elaborated again.
4. The target statement hash equals the original and its axiom certificate contains no `sorryAx`, `native_decide`, or nonstandard axiom.
5. For every node with a real proof, $\mathrm{Actual}(v) \subseteq \mathrm{Declared}(v)$. The extractor follows non-blueprint constants and stops at blueprint declarations.

Conditions 1, 2, and 5 range over the whole module. Only proof completion is restricted to C, so a disconnected deferred node does not make the target proof invalid; strict all-node completion is reported separately.

Condition 5 is an inclusion, not an equality: a repaired proof may stop using an edge that stays harmlessly declared. With DeepSeek-V4-Flash such edges remain on 10 typed, 28 patch and 6 rewrite results. On the typed side these are 12 edges, and 8 of them belong to nodes closed by the fixed node prover rather than to proofs the model wrote. We report unused edges as a quality measure and require only that no dependency stays hidden.

B.2 Stronger automation

The fixed node prover tries `rfl`, then `simp`, then `omega`, with 200,000 heartbeats for each tier; six states also declare `norm_num` as a fourth tier. On the 39 proof-authoring states we ran each of `norm_num`, `ring`, `linarith`, `nlinarith`, and `aesop` on its own at the same budget, counting only proofs the kernel accepts without extra axioms. Two states close with `aesop` and one with `nlinarith`; the other three tactics close none. Six modules needed one added import for these tactics to be in scope at all.

B.3 Outcomes by failure family

Table 7 splits the controlled benchmark by injected family. A compound state carries more than one defect, so it is counted once, in its own row, and not inside the family of its first defect. Redundant dependencies, missing dependencies and false lemmas are solved by all three interfaces almost without exception. The states that no interface solves sit in two families: 15 of the 24 are monolithic nodes and 6 are missing hypotheses.

Table 7: Solved states by failure family, for DeepSeek-V4-Flash, on the same endpoints as Table 5.

Family	n	Typed	Patch	Rewrite	None
false / too strong	18	17	16	18	0
missing hypothesis	30	21	24	21	6
missing dependency	15	15	15	15	0
redundant dependency	14	14	14	14	0
dead node	14	12	12	13	1
monolithic	31	12	12	11	15
representation	8	4	6	2	2
compound	12	9	10	10	0
total	142	104	109	104	24

B.4 Footprint of the accepted repairs

The interfaces differ in what they leave behind, on the same DeepSeek-V4-Flash endpoints as Table 5. Every accepted result of every interface is complete in the strict sense: no node is left unfinished anywhere in the module, not only inside the target closure. Rewriting most often returns a blueprint holding the target statement alone, on 42 of 104 results; typed repair does so on 38 of 104 and patching on 14 of 109. Patching leaves nodes outside the target closure on 26 results, typed repair on 19, and rewriting on 5. Patching also changes node statements most often: 62 changed statements against 16 for typed repair and 21 for rewriting.

C Experimental configuration

All model calls go through OpenRouter. The matched run uses DeepSeek-V4-Flash build `deepseek-v4-flash-20260423` (DeepSeek-AI, 2026b); the repeated run uses Qwen3.6-Flash (Qwen Team, 2026), model identifier `qwen/qwen3.6-flash`, served by a single endpoint (Alibaba). Neither run sends decoding parameters, so provider defaults apply. The maximum model output is 49,152 tokens in every arm. The three interfaces together cost $2.10 with DeepSeek-V4-Flash and $14.80 with Qwen3.6-Flash. The artifact pins the Lean/mathlib environment and records input, cached input, reasoning, and output tokens together with the provider pricing snapshot.

C.1 The prompts as sent

The three system prompts follow, verbatim, for state `p02`; across states only the namespace, the label prefix and the allowed imports differ. The typed arm also receives the ten operation schemas in the request's function-calling field, released in full with the artifact.

Typed local edits.

```
You repair formal proof blueprints (Lean 4, LeanArchitect
@[blueprint] attribute). A blueprint is a DAG of theorem nodes;
the state below FAILED for the typed reasons given. You control
the blueprint ONLY through the provided tools: each turn call
EXACTLY ONE tool; the harness applies it mechanically to the
Lean module, runs the Lean verifier and a deterministic node
prover (rfl, simp, omega on deferred nodes), and returns typed
feedback. Iterate until the target is proved or the budget runs
out.

Rules:
- The target theorem's statement is IMMUTABLE (its proof may
change).
- Proofs may be deferred with 'sorry' / 'sorry_using [deps]':
the prover tries each deferred node and proved nodes are
assembled automatically — a good architecture of simple
deferred steps wins without written proofs.
- 'axiom', 'admit', 'native_decide' are forbidden and rejected.
- A rejected tool call leaves the state unchanged but consumes
budget; the error message explains why.
- The graph in each tool result is the CURRENT state after your
edit.
Plan your edits from the typed statuses: statement_refuted
means the node is kernel-refuted (drop or replace it);
declaration_uses_sorry means the proof is deferred;
no_proof_within_budget / budget_ladder_exhausted mean the
prover failed on it as stated (split it or prove it yourself).
```

Local patching.

```
You repair formal proof blueprints (Lean 4 with the
LeanArchitect @[blueprint] attribute). A blueprint is a DAG of
theorem nodes; the state below FAILED for the typed reasons
given. You edit the CURRENT Lean module source directly with
exact search/replace patches. Each turn reply with ONE patch
attempt: one or more blocks in EXACTLY this format (only the
blocks are interpreted):

<<<<<<< SEARCH
(text that occurs in the current module source)
=======
(replacement text)
>>>>>>> REPLACE

Patch semantics (mechanical, checked before evaluation):
- SEARCH must be non-empty and must match the current module
source EXACTLY ONCE, character for character, whitespace and
line breaks included. To insert, anchor on neighboring existing
text and repeat it in the replacement; to delete, leave the
replacement side empty.
- Blocks apply in reply order, each against the text produced
by the previous block. The patch is ATOMIC: the first failing
block rejects the whole attempt and the module stays unchanged.
A rejected attempt still consumes budget; the error message
explains why.
- After a patch applies, the harness validates the module,
elaborates it with the Lean verifier, runs the deterministic
node prover on deferred nodes, and returns typed feedback plus
the UPDATED module source. Write every SEARCH against the
LATEST module source shown to you.

Rules:
- The target theorem's statement (binders and result type) must
be preserved EXACTLY; a patch whose result changes it is
rejected after application.
- 'axiom', 'admit', 'native_decide' are forbidden and rejected.
```

```
- The module must keep `namespace P02Bad` ... `end P02Bad`,
its imports within: Architect, Mathlib.Tactic.Ring,
Mathlib.Tactic.Linarith, Mathlib.Tactic.NormNum, and every
declaration a theorem carrying an @[blueprint "p02bad-..."]
annotation; node-to-node dependencies are declared with
(proofUses := [name1, name2]) on the USING node.
- Proofs may be deferred: use `:= by\n  sorry` for a lone node
or `:= by\n  sorry_using [dep1, dep2]` when the planned proof
will use those blueprint nodes. A deterministic prover (rfl,
then simp, then omega; 200000 heartbeats each) will attempt
every deferred node, and proved nodes are assembled
automatically — a good architecture of simple deferred steps
can win without any written proof.

Worked example (form only; the content is yours):

<<<<<<< SEARCH
theorem step_one (n : Nat) : n = n := by
  sorry
=======
theorem step_one (n : Nat) : n = n := by
  rfl
>>>>>>> REPLACE

Iterate until the target is proved or the budget runs out.
```

Whole-module rewriting.

```
You repair formal proof blueprints (Lean 4 with the
LeanArchitect @[blueprint] attribute). A blueprint is a DAG of
theorem nodes; the state below FAILED for the typed reasons
given. Propose a COMPLETE REPLACEMENT blueprint for the same
target: any architecture you want (any number of support lemmas
and dependencies), as long as the target theorem statement is
preserved exactly.

You will iterate: after each attempt the harness elaborates
your module with
the Lean verifier, runs the deterministic node prover (rfl,
simp, omega) on
deferred nodes, and returns feedback. Reply to feedback with
the COMPLETE
corrected module (one lean fence, full contract), not a
description.

Output contract (mechanical, checked before evaluation):
- Reply with EXACTLY ONE fenced code block labeled lean
containing a complete Lean 4 module; no other code blocks.
- The module imports must be a subset of: Architect,
Mathlib.Tactic.Ring, Mathlib.Tactic.Linarith,
Mathlib.Tactic.NormNum.
- The module must declare `namespace P02Bad` and `end P02Bad`.
- Every blueprint node is a theorem carrying the attribute
  @[blueprint "<label>" (latexEnv := "lemma")] for support
lemmas or
  @[blueprint "<label>"] for the target theorem, where every
<label> starts with "p02bad-" (for example "p02bad-target").
- Declare node-to-node dependencies in the attribute with
(proofUses := [name1, name2]) on the USING node; dependencies
not declared there are invisible to the graph.
- The target theorem's statement text (binders and result type)
must be preserved EXACTLY as given; changing it invalidates the
candidate.
- Proofs may be deferred: use `:= by\n  sorry` for a lone node
or `:= by\n  sorry_using [dep1, dep2]` when the planned proof
will use those blueprint nodes. A deterministic prover (rfl,
then simp, then omega; 200000 heartbeats each) will attempt
every deferred node, and proved nodes are assembled
automatically — a good architecture of simple deferred steps
can win without any written proof.
- You may also write complete proofs using core Lean lemmas and
the allowed imports.

Follow this reply template EXACTLY (structure and syntax;
content is yours;
the fence label is `lean`, and `import Architect` must be the
first import
— the blueprint attribute lives there):

action: <action name>

```lean
import Architect

namespace P02Bad

@[blueprint "p02bad-step-one"
  (latexEnv := "lemma")]
theorem step_one (n : Nat) : n = n := by
  sorry

@[blueprint "p02bad-target"
  (proofUses := [step_one])]
theorem target_name (n : Nat) : n = n := by
  sorry_using [step_one]

end P02Bad
```
```

D Released trace fields

Each BLUEPRINTTRACE episode records the protocol configuration; initial state and target signature; canonical source, node statements, statuses, and edges at every step; the typed call, patch blocks, or rewritten module; whether the action was applied and its source/graph difference; Lean, node-prover, and graph-scan feedback; every rejection reason; token and cost usage; and the final outcome. Controlled states also include source-target provenance, construction metadata, and an available correct source blueprint or defect manifest; the ten function schemas ship as one machine-readable file.

From Recognition to Reconstruction: Towards Robust Procedural Mathematical Reasoning

Fateme Mazdarani and Carlos Toxtli
School of Computing, Clemson University
Clemson, SC, USA
fmazdar@clemson.edu, ctoxtli@clemson.edu

Abstract

Mathematical reasoning systems should be consistent when the same concept is expressed in equivalent forms. This requires not only recognizing shared mathematical meaning but also identifying the representation changes and reconstructing a traceable route between forms. We study this problem through mathematical theorems, whose equivalent formulations preserve the same underlying concept while differing substantially in representation. In this setting, theorem recognition is only the first step: a robust system must also identify the intervening transformations and recover their order. We introduce an evaluation of robust procedural mathematical reasoning with three components: theorem identification, transformation identification, and ordered reconstruction. Using validated theorem–representation pairs from an existing corpus, we build a procedural-robustness dataset with typed operations, reference procedures, and matched contrasts that distinguish valid reformulations from invalid ones. We first measure closed-book performance of four open-weight models to establish a baseline, then evaluate retrieval-augmented generation (RAG) as support mechanism for the same task. Models identify theorem identity more reliably than they recover the transformations and ordered routes connecting equivalent representations. RAG improves every stage, but procedural reconstruction remains the main bottleneck. The framework aims to inspire more consistent and traceable mathematical systems, with potential applications in mathematical search, tutoring, autoformalization, and scientific discovery.

1 Introduction

The ability to reason reliably across changes in representation is fundamental to intelligent systems. One concept may be expressed through different symbols, structures, abstractions, or intermediate constructions, yet these surface differences should not change a system's conclusions. A robust system should recognize the shared meaning, identify how the representation has changed, and reconstruct a traceable connection between the different forms.

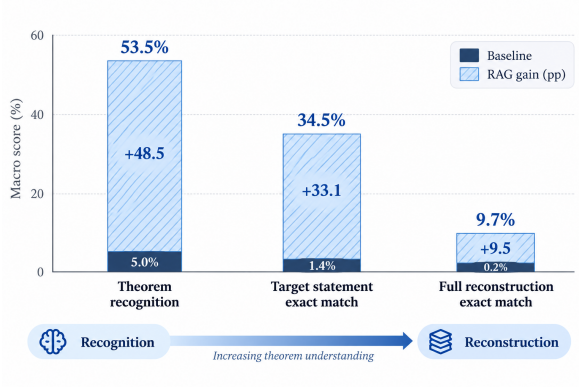


Figure 1: From recognition to reconstruction: macro-averaged scores across four models under increasingly strict evaluation criteria. Solid segments show baseline scores; hatched segments show RAG gains in percentage points. Bar-top labels report RAG scores.

Mathematical theorems provide a particularly demanding setting for studying this capability. A theorem may appear as an equation, a set relation, an optimization problem, or an auxiliary construction without changing its underlying content. Robustness therefore cannot be established through success on a single familiar notation. We formulate the broader capability as *robust procedural mathematical reasoning*. Given a reformulated theorem statement, a model must identify the original theorem, determine the transformations connecting the observed and reference representations, and order those transformations into a reconstruction route that reaches the reference statement. These are distinct capabilities: a model may correctly name the theorem while omitting a required operation, adding an unnecessary step, or placing the correct transformations in the wrong order.

This capability is relevant to mathematical search, tutoring, and autoformalization, where sys-

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 42–56
October 29, 2026 ©2026 Association for Computational Linguistics

tems must connect differently expressed statements while exposing the transformations that support the match (Yang et al., 2023; Wu et al., 2022). It may also make AI-assisted mathematical discovery easier to inspect and verify (Romera-Paredes et al., 2024; Trinh et al., 2024). Existing benchmarks commonly evaluate final-answer correctness in mathematical problem solving (Cobbe et al., 2021; Hendrycks et al., 2021) or success in formal proof generation (Polu and Sutskever, 2020; Zheng et al., 2021). These outcomes are important, but they do not directly show whether a model can explain how two equivalent representations are connected. A model may identify the correct theorem from surface cues while failing to recover the mathematical changes that preserve equivalence. Evaluating the reconstruction process can therefore reveal errors hidden by theorem-recognition accuracy alone and provide a more traceable account of model behavior. The TREAT dataset provides validated non-canonical reformulations of named theorems for evaluating recognition across representations (Mazdarani and Toxtli, 2026b). Yet, it does not evaluate the recovery process itself.

To support procedural evaluation, and address the gap, we extend a stratified subset of TREAT for procedural evaluation. We introduce a fixed ontology of transformation operations, assign typed labels to the changes required by each example, and provide an ordered reference route from the reformulated statement to the reference form. We also add matched contrasts that retain the overall representation while changing a mathematically decisive detail. The resulting procedural-robustness benchmark separates theorem identification from transformation identification, ordered reconstruction, and sensitivity to local validity.

Our evaluation proceeds in two stages. We first assess four open-weight models to establish their baseline performance on each component of the task. We then evaluate the same models using retrieval-augmented generation (RAG) under three conditions that provide theorem cards, transformation cards, or both. The theorem cards supply information about candidate theorems and their assumptions, while the transformation cards describe the operations used to connect different representations. This design allows us to test whether external theorem knowledge, procedural guidance, or their combination can address the weaknesses observed without retrieval. These gaps and design choices motivate the following research questions:

RQ1: How reliably do models identify the underlying theorem when it is expressed through a non-canonical but equivalent representation?

RQ2: How accurately do models identify the required transformations and reconstruct a route to the reference statement?

RQ3: To what extent does RAG improve theorem identification and procedural reconstruction?

RQ4: How does performance vary across representation mechanisms, route lengths, mathematical families, models, and matched validity contrasts?

2 Related Work

2.1 Formal knowledge and representation robustness

Language models can reproduce relational knowledge from pretraining (Petroni et al., 2019), but access remains sensitive to query form. ParaRel finds inconsistent factual predictions across paraphrases (Elazar et al., 2021), while negated and misleading probes expose failures on locally decisive relations (Kassner and Schütze, 2020). These results motivate testing whether access to the same object remains stable when its representation changes.

Mathematical benchmarks similarly vary surface form, symbols, templates, and related inputs (Zhou et al., 2024a; Mirzadeh et al., 2024; Zhou et al., 2024b). MATH Perturb and PutnamGAP extend controlled perturbation to advanced problems (Huang et al., 2025; Hao et al., 2025). TREAT pairs named theorems with equivalent formula-level representations and evaluates theorem identification (Mazdarani and Toxtli, 2026b). We use a stratified subset to ask the procedural question left open by recognition: can models reconstruct the representation changes linking an observed formulation to a reference statement?

2.2 Procedural mathematical reasoning

Most mathematical language-model evaluations emphasize answers, proofs, or formal derivations. TheoremQA studies theorem application (Chen et al., 2023a); NaturalProofs and NaturalProver evaluate grounded proof generation (Welleck et al., 2021, 2022); and autoformalization and theorem-proving systems operate in proof-assistant environments (Wu et al., 2022; Yang et al., 2023; Jiang et al., 2022). Process supervision and error localization further separate correct answers from correct derivations (Lightman et al., 2023; Zheng et al., 2024; Mazdarani and Toxtli, 2026a). Our

reconstruction sequence is not a theorem proof. It is a typed, representation-level account of how an equivalent statement maps to a reference statement under recorded assumptions. It connects to expression transformation (Edelmann and Kuncak, 2019) and operator-based formal tactics (Samoylov and Vosoughi, 2025), but targets traceable reconstruction of a known object rather than proof search.

2.3 RAG and structured-output evaluation

Retrieval-augmented generation conditions generation on retrieved context (Lewis et al., 2020). In mathematics, retrieved premises, theorems, and proofs support natural-language and formal proving (Welleck et al., 2022; Yang et al., 2023, 2025), while formula-text retrieval links symbolic expressions with explanatory context (Li, 2025). Here, the observed statement is the query, a deterministic retriever returns theorem or transformation cards, and the same model generates a structured reconstruction. RAG is therefore an enabling condition, not the object of optimization. Structured-prediction evaluation distinguishes complete-output exact match from substructure overlap (Yu et al., 2018; Chen et al., 2023b). We score statements and ordered procedures by exact match, transformation content by multiplicity-aware label F1 (Yang et al., 2019), and paired decisions by contrast-set consistency (Gardner et al., 2020; Johnson and Marasović, 2023). Execution-based evaluation is unavailable because the procedures do not share an executable formalism (Zhong et al., 2020).

3 Methods

3.1 Problem Formulation

Let y denote a theorem identity and r_y its designated reference statement. The source corpus provides a reformulation x validated as equivalent to r_y under recorded assumptions. We annotate a reference procedure

$$x = x_0 \xrightarrow{o_1} x_1 \xrightarrow{o_2} \cdots \xrightarrow{o_L} x_L = r_y, \quad (1)$$

where each o_j is a typed, item-specific representation rewrite. Given x and optional RAG context, the model returns ACCEPT, REJECT, or ABSTAIN. An accepted output contains a theorem prediction $\hat{y}$, reconstructed statement $\hat{r}$, and ordered operation sequence $\hat{\tau}$. These fields evaluate theorem identification, transformation identification, and ordered reconstruction. The reference statement r_y is the formulation selected by the source corpus as the common formulation. It is not assumed to be the unique form of the theorem. Similarly, the annotated procedure records one valid sequence of representation changes from x to r_y; other valid sequences may also exist. The procedure is intended for evaluating reconstruction and should not be interpreted as a proof of the theorem. For example, from $\delta_D(\chi'_l(K_{n,n}), n) = 0$, a successful output identifies Dinitz's theorem, applies distance removal, and reconstructs $\chi'_l(K_{n,n}) = n$. Appendix A.3 provides more examples.

3.2 Dataset Construction

TREAT contains 29,480 validated reformulations of 737 named results, with source and transformed statements, assumptions, metadata, and recovery descriptions (Mazdarani and Toxtli, 2026b). We select 1,400 reformulations from 306 theorem identities and map their recovery descriptions to a fixed operation ontology. Each procedure is oriented from the observed statement to the designated reference statement and split into atomic steps that record an operation identifier and its input and output expressions. Each reformulation is linked to its theorem identity, reference statement, transformation labels, and ordered reference procedure. The ontology contains 16 operations. Fourteen remove representation mechanisms, including distance encodings, set membership, optimization forms, and ordered pairs, while two normalize notation or relation direction. The full inventory appears in Appendix A.1. Each reformulation is also paired with a matched contrast that preserves the outer representation while changing one locally decisive component (Gardner et al., 2020). The four contrast types are target-expression perturbation, logical negation, relation-direction reversal, and type-matched replacement, with 350 examples of each (See Appendix A.1, Table 5).

The selected reformulations are stratified across five mathematical families and 20 procedure signatures. Each family contributes 280 items, each signature contributes 70, and no theorem contributes more than five. The reference procedures contain one, two, or three scored operations, corresponding to 980, 280, and 140 items, respectively. The full dataset contains 2,800 inputs: 1,400 reformulations and 1,400 matched contrasts. During evaluation, models receive only an opaque item identifier and the observed statement.

3.3 Experimental Design

We evaluate procedural reconstruction under four experimental conditions. The *baseline condition* uses the common task prompt, structured output contract, and operation inventory, without additional theorem or transformation evidence. We then evaluate whether retrieval-augmented generation (RAG) improves performance under three supported conditions: theorem RAG, transformation RAG, and combined RAG. The *theorem-RAG condition* provides cards containing theorem names, aliases, reference statements, assumptions, and domain metadata. The *transformation-RAG condition* provides cards describing reconstruction operations, applicability conditions, input and output types, and worked examples. The *combined-RAG condition* provides both types of cards. Comparing these conditions with the baseline allows us to measure the effects of theorem information, transformation guidance, and their combination.

RAG is implemented as a transparent and deterministic support mechanism because the experiment examines how external evidence affects procedural reconstruction rather than how retrieval itself can be optimized. For query q and document d, the retrieval score is

$$s(q,d) = \frac{|T(q) \cap T(d)|}{\sqrt{|T(d)|}}, \tag{2}$$

where $T(\cdot)$ contains lowercase alphanumeric tokens longer than one character. This reproducible lexical score requires no fitted corpus statistics (Robertson and Zaragoza, 2009). Each active index returns its top eight cards, while all other experimental settings remain fixed.

3.4 Models and inference

We evaluate GPT-OSS-20B, GPT-OSS-120B (OpenAI, 2025), GEMMA-4-31B (Google DeepMind, 2026), and LEANSTRAL-2603 (Mistral AI, 2026). Each model processes all 2,800 inputs in all four settings, yielding 44,800 predictions. Decoding uses temperature 0, top-$p = 1$, seed 20260723, and a 2,200-token output limit.

Raw responses are retained before parsing. A logged deterministic parser applies predefined formatting repairs, normalizes field names, and maps out-of-vocabulary transformation strings to `OTHER`. It does not alter mathematical expressions, theorem predictions, or substantive reconstruction content. The 1,045 outputs (2%) that remain invalid after repair stay in the denominator and receive zero task credit. All intended records are present and no backend failures occurred. The shared system prompt, user template, and structured output contract are provided in Appendix A.1.

3.5 Evaluation Metrics

Each model output contains a decision, theorem prediction, reconstructed statement, and ordered sequence of transformation labels. We use exact-match metrics to evaluate fully specified outputs and an F1 score to measure partial recovery of the required transformations. This follows the general structured-prediction principle of evaluating both the complete output and its recoverable components (Chen et al., 2023b).

3.5.1 Theorem and Reconstruction Metrics

Theorem recognition accuracy requires an ACCEPT decision and the correct theorem identity after alias normalization.

Target statement exact match additionally requires the reconstructed statement to match the designated reference statement r_y after deterministic presentation normalization. This metric evaluates recovery of the designated target statement independently of the predicted operation sequence.

Transformation-label F1 measures order-invariant overlap between the predicted and reference transformation labels. Let $m_o(\tau)$ denote the multiplicity of operation o in procedure τ, and define

$$M(\hat{\tau}, \tau) = \sum_{o} \min!\left(m_o(\hat{\tau}), m_o(\tau)\right). \tag{3}$$

Precision is $M/|\hat{\tau}|$, recall is $M/|\tau|$, and their harmonic mean is reported. An empty predicted sequence receives a score of zero. The comparison preserves multiplicity but ignores order. The score is gated by theorem correctness because the same operation labels may apply to different theorems. It gives partial credit for correctly recovered labels despite omissions, additions, or reordering, but does not establish that the predicted sequence is mathematically valid.

Full reconstruction exact match requires an ACCEPT decision, the correct theorem, an exact match with the target statement, and exact agreement with the complete ordered reference procedure. This is the strictest metric; each item contains one annotated reference procedure rather than

every possible valid route, the metric measures reproducible agreement with the annotation rather than uniqueness of the mathematical path.

By design, *full reconstruction exact match* cannot exceed *target statement exact match*, which cannot exceed *theorem recognition accuracy*. This ordering is therefore expected. The informative quantities are the sizes of the gaps between these metrics and how those gaps change from the baseline condition to the RAG-supported conditions.

3.5.2 Matched-Contrast Metrics

Contrast false-positive rate is the percentage of contrasts assigned ACCEPT.

Contrast-set consistency is the percentage of matched pairs for which the valid reformulation is accepted and its contrast is rejected. Contrast-set consistency is stricter than item-level classification accuracy because both decisions must be correct within the same pair.

3.5.3 Retrieval and Statistical Analysis

We evaluate RAG's retrieval metrics separately from task performance; *Theorem recall at* k records whether the gold theorem card appears among the top k retrieved results. *Transformation coverage* records whether retrieval contains at least one required transformation and whether it contains the complete required set. *Joint coverage* requires retrieval of both the gold theorem card and every transformation in the reference procedure.

All task metrics are first computed separately for each model. Aggregate results are macro-averaged across models. Primary differences between the baseline and RAG-supported conditions are estimated using 10,000-repetition hierarchical bootstrap resampling over models and theorem clusters or matched pairs (Saravanan et al., 2020). We report absolute percentage-point differences with 95% percentile intervals.

4 Results

4.1 Baseline Capability and the Reconstruction Gap

Table 1 reports the performance of each model under the baseline and three RAG-supported conditions. Under the baseline condition, macro performance is low across all positive-input metrics, indicating limited ability to identify the underlying theorem and reconstruct the route from a reformulated statement to its reference form. Theorem

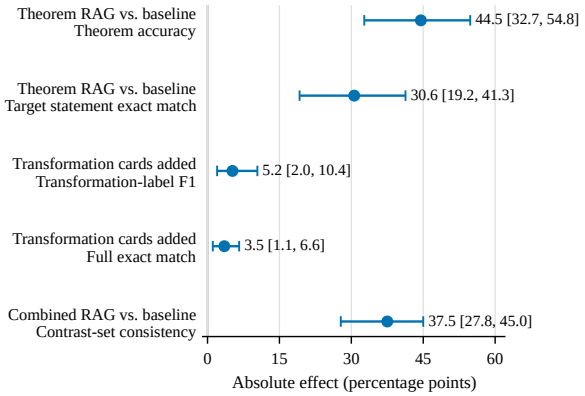


Figure 2: Primary RAG condition effects with 95% hierarchical bootstrap intervals. The first two rows compare theorem RAG with the baseline condition. The next two compare combined RAG with theorem RAG, isolating the effect of adding transformation cards. The final row compares combined RAG with the baseline condition.

recognition accuracy exceeds target statement exact match, which in turn exceeds full reconstruction exact match. Although this ordering follows from the nested definitions of the metrics, the size of the declines is informative. Models may identify the theorem without recovering the designated target statement, and they may recover the target without reproducing the complete ordered procedure. Table 7 in Appendix A.2 decomposes combined-RAG outcomes into mutually exclusive success and failure categories.

4.2 Effects of RAG Support and Retrieval Coverage

The RAG conditions improve performance over the baseline, but the gains differ across stages of the task. Theorem-card RAG produces large gain in theorem identification: relative to the baseline condition, theorem recognition accuracy increases from 5.0% to 49.5%. Transformation cards provide complementary procedural support. Adding transformation cards to theorem RAG increases transformation-label F1 from 13.7% to 18.9%. Figure 2 demonstrate these effects.

Combined RAG produces the strongest macro-level results, reaching 53.5% theorem recognition accuracy, 34.5% target statement exact match, 18.9% transformation-label F1, and 9.7% full reconstruction exact match. Thus, retrieval provides substantial support for identifying the theorem and recovering the target statement, but considerably less support for reconstructing the complete ordered route.

Model	RAG setting	Thm. Acc.	Target EM	Trans. F1	Full EM	Contrast Cons.	Contrast FPR
gpt-oss-20B	No RAG	5.6	1.4	1.2	0.1	17.4	0.9
	Theorem RAG	48.6	30.8	6.7	2.4	46.9	2.1
	Transform. RAG	6.2	1.9	2.3	0.6	28.3	2.4
	Combined RAG	56.9	35.1	19.5	10.6	54.9	2.0
gpt-oss-120B	No RAG	8.4	2.1	2.5	0.2	19.6	1.6
	Theorem RAG	65.4	35.4	20.1	7.0	60.0	6.1
	Transform. RAG	8.5	2.0	3.1	0.9	29.6	3.9
	Combined RAG	67.9	36.0	23.5	9.9	64.9	4.6
Gemma-4-31B	No RAG	5.8	2.1	2.2	0.5	8.9	0.9
	Theorem RAG	57.1	47.1	20.2	11.9	51.6	5.9
	Transform. RAG	11.1	3.5	4.0	1.1	18.3	3.1
	Combined RAG	58.9	49.7	22.4	14.1	53.1	7.4
Leanstral-2603	No RAG	0.1	0.0	0.0	0.0	0.2	0.0
	Theorem RAG	27.1	14.5	7.7	3.3	21.4	5.7
	Transform. RAG	0.2	0.0	0.0	0.0	0.6	0.0
	Combined RAG	30.1	17.1	10.2	4.1	23.1	6.9
Macro average	No RAG	5.0	1.4	1.5	0.2	11.5	0.8
	Theorem RAG	49.5	32.0	13.7	6.1	45.0	5.0
	Transform. RAG	6.5	1.8	2.4	0.6	19.2	2.4
	Combined RAG	**53.5**	**34.5**	**18.9**	**9.7**	**49.0**	**5.2**

Table 1: Results for each model and experimental condition (%). The baseline condition provides no retrieved evidence, while the three RAG conditions provide theorem cards, transformation cards, or both. Macro averages are reported in the final four rows. Thm. Acc. denotes theorem recognition accuracy; Target EM, target statement exact match; Trans. F1, transformation-label F1; Full EM, exact agreement on the theorem, target statement, and complete ordered reference procedure; Contrast Cons., contrast-set consistency; and Contrast FPR, the false-positive rate on matched contrasts.

4.3 Retrieval Coverage Analysis

The retrieval coverage analysis separates failures to retrieve relevant evidence from failures to use available evidence. Under combined RAG, the gold theorem card appears among the top eight results for 76.4% of valid inputs. The complete set of required transformations is retrieved for 57.6%, while both the gold theorem and every required transformation are available for 42.9% of inputs, as shown in Table 2.

On the 601 examples with complete retrieved coverage, theorem recognition accuracy reaches 74.5%, target statement exact match reaches 49.2%, transformation-label F1 reaches 41.9%, and full reconstruction exact match reaches 22.0%. These are coverage-conditioned results on items for which the retriever succeeds, not oracle estimates for the full dataset. They nevertheless show that relevant RAG context can be used productively while leaving a substantial challenge in selecting, applying, and ordering the available information.

4.4 Performance Across Reconstruction Structures

Reconstruction difficulty increases with procedure length. Under combined RAG, full reconstruction exact match is 13.8% for one-operation procedures and zero for two- and three-operation procedures (Table 3), while theorem recognition, target recovery, and transformation-label F1 remain nonzero. However, the length groups are not matched for theorem identity or representation mechanism. They contain 295, 139, and 73 distinct theorem identities, respectively. One-operation procedures cover all 14 representation mechanisms; two-operation procedures cover distance, containment, set membership, and truth-table encodings; three-operation procedures cover only distance and set membership. Although the five mathematical families are equally represented within each length group, this balance does not control theorem or mechanism composition. These differences could partly explain the observed performance decline. The length breakdown therefore describes performance across the sampled procedure groups rather than isolating the effect of adding reconstruction steps.

Performance also differs across representation mechanisms. Distance, argmin, witness, and quantifier encodings show stronger transformation recovery. In contrast, truth-table, fiber, cylinder, and ordered-pair encodings often lead models to identify the correct theorem without predicting the annotated transformation sequence. In this balanced sample, variation across representation mechanisms is greater than variation across broad mathematical families. Appendix A.2 provides the length and mechanism profiles (Figures 3 and 4) and family-level results (Table 6).

The matched contrasts measure whether model decisions remain sensitive to locally decisive

(a) RAG retrieval coverage (%)

Gold card coverage	R@1	R@4	R@8
Theorem card	60.0	72.4	76.4
At least one transformation card	n/a	n/a	69.1
All required transformation cards	n/a	n/a	57.6
Theorem and all transformation cards	n/a	n/a	42.9

(b) Combined RAG conditioned on coverage (%)

Retrieved gold context	n	Thm. Acc.	Target EM	Trans. F1	Full EM
Neither	125	0.0	0.0	0.0	0.0
Transformations only	205	0.0	0.0	0.0	0.0
Theorem only	469	64.2	39.8	2.7	0.6
Complete	601	74.5	49.2	41.9	22.0

Table 2: RAG retrieval coverage and generation performance conditioned on coverage. The count in panel (b) is the number of unique valid inputs.

L	n	Thm. Acc.	Target EM	Trans. F1	Full EM
1	980	55.8	36.4	23.0	13.8
2	280	48.4	31.7	9.2	0.0
3	140	47.1	26.6	9.8	0.0

Table 3: Combined RAG by procedure length (%).

changes. Combined RAG increases contrast-set consistency from 11.5% to 49.0%, while the contrast false-positive rate increases from 0.8% to just 5.2%. Logical negation and relation-direction reversal are the most difficult contrast types, whereas target-expression perturbations and type-matched replacements are rejected more reliably.

5 Discussion

The results reveal a clear separation between theorem recognition and procedural reconstruction. Retrieval substantially improves access to the underlying theorem and reference statement, but complete recovery of the ordered transformation route remains difficult. The matched contrasts further show that consistency across valid reformulations must be balanced with sensitivity to small changes that invalidate the theorem mapping.

5.1 RQ1: Theorem identification is necessary but not sufficient

The baseline results show that non-canonical representations make even familiar theorem knowledge difficult to access. Theorem RAG substantially improves recognition, indicating that many reformulated statements can be linked to the correct mathematical concept when relevant information is available. The remaining gap between theorem recognition, target recovery, and full reconstruction is more important than the improvement in recognition alone. Identifying the theorem establishes what the observed statement represents, but it does not explain how that statement relates to the designated reference form. Models often recover the conceptual destination without recovering the transformations needed to reach it. This distinction persists in the model-level results, where performance rankings change across theorem identification, target recovery, reconstruction, and contrast consistency. Theorem recognition is therefore a necessary first step, but it is not a sufficient measure of robustness across mathematical representations.

5.2 RQ2: Route Recovery Is the Main Bottleneck

Models may recover the reference statement without recovering the complete sequence of transformations that connects it to the observed form. Transformation-label F1 shows that many failed exact reconstructions still contain partial procedural information: models often identify the main operation but omit a normalization step, add an unnecessary operation, or predict the wrong order. Longer-procedure groups exhibit lower exact-match performance, although this comparison is confounded by differences in theorem and mechanism composition. Table 3 shows that full reconstruction exact match falls to zero for two- and three-operation procedures, although theorem recognition, target recovery, and transformation-label F1 remain nonzero. Models therefore recover parts of longer routes but do not reliably reproduce every required step.

Performance also varies across mechanisms, suggesting that some reformulations make their inverse operations more visible than others. Detailed results appear in Appendix A.2, although the present evaluation cannot separate structural difficulty from model familiarity. Because each item contains one annotated reference route, full reconstruction exact match measures agreement with a reproducible scoring target rather than the uniqueness of the mathematical path. It therefore distinguishes reaching the correct endpoint from recovering a complete and inspectable route.

5.3 RQ3: RAG Supports but Does Not Solve Procedural Reconstruction

The retrieval sources provide complementary support. Theorem cards mainly improve theorem identification and target recovery, while transformation cards provide information about the operations connecting the observed and reference forms. Transformation cards are useful once theorem information is available, indicating that procedural guidance depends on a plausible theorem-level interpretation. Retrieval coverage separates failures to find relevant evidence from failures to use it. Table 2 shows higher performance when the gold theorem card and all required transformations are retrieved, yet full reconstruction remains difficult even under complete coverage. Access to the necessary evidence therefore does not guarantee correct selection, application, or ordering. Retrieval and reconstruction should consequently be treated as separate components. Retrieval can narrow the space of plausible theorems and transformations, but future work must determine whether the generated steps compose into a mathematically valid route.

5.4 RQ4: Robustness Requires Global Recognition and Local Validity

The matched contrasts test whether models remain selective when a reformulation is changed in a mathematically decisive way. Combined RAG improves contrast-set consistency, showing that retrieval helps models accept valid reformulations while rejecting their paired contrasts. However, the accompanying increase in false positives indicates that stronger theorem-level evidence can also make models more willing to accept invalid variants.

The contrast-type breakdown in Table 8 shows that logical negation and relation-direction reversal are the most difficult edits. These changes preserve most of the surrounding expression while altering a relation that determines whether the original theorem and reconstruction route remain applicable. Target-expression perturbations and type-matched replacements are rejected more reliably because they produce more visible changes to the mathematical structure.

This pattern highlights the need to combine global recognition with local validation. Identifying a strong match to a known theorem is not sufficient when a negated event, reversed inequality, or changed inclusion invalidates the connection. A robust system should therefore verify the locally decisive relations before accepting the theorem identity and proposed reconstruction route.

5.5 Broader Implications and Applications

The present results only establish benchmark behavior, yet inspires future studies. Procedural reconstruction can make mathematical systems more traceable by showing how an unfamiliar representation connects to a known result. In mathematical search, it can explain why a theorem expressed in different notation or structure is relevant. In tutoring and assessment, separating theorem identification, transformation recovery, and operation ordering can localize the source of an error. The same framework may support autoformalization, proof repair, and theorem proving by exposing the transformations needed to align nonstandard inputs with formal reference statements. It also motivates modular systems in which retrieval proposes relevant knowledge, reconstruction builds the route, and a symbolic or formal checker verifies each step. More broadly, exposing the transformations behind a proposed connection may make mathematical and scientific discovery more inspectable and testable than returning only a label or analogy (Udrescu and Tegmark, 2020; Romera-Paredes et al., 2024; Trinh et al., 2024).

6 Conclusion

We introduced robust procedural mathematical reasoning as the task of identifying a theorem, recovering the transformations between equivalent representations, and ordering them into a traceable route to a reference statement. We extended a stratified subset of validated theorem reformulations with typed operations, ordered reference procedures, and matched contrasts.

The results show that theorem recognition is substantially easier than procedural reconstruction. Retrieval-augmented generation improves theorem identification, target recovery, and transformation recovery, but complete reconstruction remains difficult even when the relevant evidence is retrieved. Reliable representation robustness therefore requires not only access to the correct theorem, but also the ability to select, apply, and order locally valid transformations.

Acknowledgment

This work was supported in part by the National Science Foundation under Award No. 2434704.

Limitations

This study provides a controlled evaluation of procedural reconstruction, and its conclusions are scoped to the benchmark, annotations, metrics, and experimental setup used here.

Each reformulation has one designated reference statement and one annotated reconstruction procedure, although other mathematically valid routes may exist. Full reconstruction exact match may therefore undercount valid alternatives, while transformation-label F1 ignores order and does not verify that the predicted operations compose into a valid route. Target statement exact match handles presentation-level variation but not unrestricted symbolic equivalence. Moreover, the procedures are not executable in a shared formal system, so operation preconditions, intermediate statements, and the use of theorem assumptions are not independently verified. Additional expert annotation or formal validation would strengthen these aspects.

The benchmark covers a stratified subset of TREAT, five mathematical families, 16 operations, and procedures of one to three steps, with fewer examples at longer lengths. The matched contrasts contain four controlled single-edit types. Consequently, the results may not extend to longer procedures, other mathematical domains, alternative transformation ontologies, multiple interacting edits, or failures involving implicit assumptions and domain restrictions. The subgroup analyses are descriptive and cannot fully separate structural difficulty from theorem familiarity or possible pretraining exposure. A human or expert baseline would also help calibrate the benchmark's absolute difficulty.

The experiments evaluate four open-weight models using one prompt, structured output schema, deterministic decoding configuration, and parser. The reported model averages should therefore not be interpreted as estimates of performance across the broader model population, and different prompting or decoding choices may change the results.

Finally, the RAG conditions use a reproducible lexical retriever rather than an optimized retrieval system. The theorem cards intentionally provide reference statements, so these conditions measure models' ability to use supplied evidence rather than independently recover all information. The effects of individual card fields are not isolated. Coverage-conditioned results are also associational, since examples with complete retrieval may be systematically easier. Future work should examine card ablations, stronger retrievers, broader model families, human evaluation, and symbolic or formal checking of reconstruction steps.

References

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023a. TheoremQA: A theorem-driven question answering dataset. *arXiv preprint arXiv:2305.12524*.

Yunmo Chen, William Gantt, Tongfei Chen, Aaron Steven White, and Benjamin Van Durme. 2023b. A unified view of evaluation metrics for structured prediction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12868–12882. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Łukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Romain Edelmann and Viktor Kuncak. 2019. Neural-network guided expression transformation. *arXiv preprint arXiv:1902.02194*.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, and 7 others. 2020. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323. Association for Computational Linguistics.

Google DeepMind. 2026. Gemma 4 model card. Google AI for Developers.

Yuren Hao, Xiang Wan, and Chengxiang Zhai. 2025. An investigation of robustness of LLMs in mathematical reasoning: Benchmarking with mathematically-equivalent transformation of advanced mathematical problems. *arXiv preprint arXiv:2508.08833*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle Cai, Hui Yuan, Runzhe Wang, Yue Wu, Ming Yin, Shange Tang, Yangsibo Huang, Chi Jin, Xinyun Chen, Chiyuan Zhang, and Mengdi Wang. 2025. MATH-Perturb: Benchmarking LLMs' math reasoning abilities against hard perturbations. *arXiv preprint arXiv:2502.06453*.

Albert Q. Jiang, Wenda Li, Szymon Tworkowski, Konrad Czechowski, Tomasz Odrzygozdz, Piotr Milos, Yuhuai Wu, and Mateja Jamnik. 2022. Thor: Wielding hammers to integrate language models and automated theorem provers. *arXiv preprint arXiv:2205.10893*.

Jacob K. Johnson and Ana Marasović. 2023. How much consistency is your accuracy worth? In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 250–260. Association for Computational Linguistics.

Nora Kassner and Hinrich Schütze. 2020. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7811–7818. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rocktaschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*.

Zichao Li. 2025. Formula-text cross-retrieval: A benchmarking study of dense embedding methods for mathematical information retrieval. In *Proceedings of the 3rd Workshop on Mathematical Natural Language Processing*, pages 124–133. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Fateme Mazdarani and Carlos Toxtli. 2026a. Beyond the answer key: Robustness evaluation of large language models for step-level mathematical verification. In *Proceedings of the 2026 IEEE International Conference on Machine Learning and Applications (ICMLA)*.

Fateme Mazdarani and Carlos Toxtli. 2026b. TREAT: Evaluating access to formal knowledge across equivalent mathematical representations. In *Proceedings of the 2026 International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.

Mistral AI. 2026. Leanstral: Open-source foundation for trustworthy vibe-coding. Mistral AI technical release.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card. OpenAI technical report.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2463–2473. Association for Computational Linguistics.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. 2024. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475.

Elisaveta Samoylov and Soroush Vosoughi. 2025. Modeling tactics as operators: Effect-grounded representations for Lean theorem proving. In *Proceedings of the 3rd Workshop on Mathematical Natural Language Processing*, pages 168–175. Association for Computational Linguistics.

Varun Saravanan, Gordon J. Berman, and Samuel J. Sober. 2020. Application of the hierarchical bootstrap to multi-level data in neuroscience. *Neurons, Behavior, Data Analysis, and Theory*, 3(5):1–25.

Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. 2024. Solving olympiad geometry without human demonstrations. *Nature*, 625:476–482.

Silviu-Marian Udrescu and Max Tegmark. 2020. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631.

Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. 2021. NaturalProofs: Mathematical theorem proving in natural language. *arXiv preprint arXiv:2104.01112*.

Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. 2022. NaturalProver: Grounded mathematical proof generation with language models. *arXiv preprint arXiv:2205.12910*.

Yuhuai Wu, Albert Q. Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. *arXiv preprint arXiv:2205.12615*.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. LeanDojo: Theorem proving with retrieval-augmented language models. *arXiv preprint arXiv:2306.15626*.

Pengcheng Yang, Fuli Luo, Shuming Ma, Junyang Lin, and Xu Sun. 2019. A deep reinforced sequence-to-set model for multi-label classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5252–5258. Association for Computational Linguistics.

Wenjie Yang, Ruiyuan Huang, Jiaxing Guo, Zicheng Lyu, Tongshan Xu, Shengzhong Zhang, Lun Du, Da Zheng, and Zengfeng Huang. 2025. Retrieval-augmented language models are mimetic theorem provers. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 21301–21313. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921. Association for Computational Linguistics.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. ProcessBench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2021. miniF2F: A cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*.

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-SQL with distilled test suites. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 396–411. Association for Computational Linguistics.

Yue Zhou, Yada Zhu, Diego Antognini, Yoon Kim, and Yang Zhang. 2024a. Paraphrase and solve: Exploring and exploiting the impact of surface form on mathematical reasoning in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2804. Association for Computational Linguistics.

Zihao Zhou, Shudong Liu, Maizhen Ning, Wei Liu, Jindong Wang, Derek F. Wong, Xiaowei Huang, Qiufeng Wang, and Kaizhu Huang. 2024b. Is your model really a good math reasoner? evaluating mathematical reasoning with checklist. *arXiv preprint arXiv:2407.08733*.

A Reproducibility and Additional Results

A.1 Prompt and Outputs

All models and experimental conditions use the same system prompt, user template, structured output contract, and fixed operation inventory. The model receives the observed statement, the active evidence lists, and the available transformation identifiers, and returns one JSON object containing a decision, theorem prediction, reference statement, reconstruction sequence, and supporting evidence identifiers.

Listing 1: User prompt template. Bracketed fields are populated at inference time.

```
OBSERVED_STATEMENT_LATEX:
{{observed_statement_latex}}

THEOREM_EVIDENCE:
{{theorem_evidence_or_NONE}}

TRANSFORMATION_EVIDENCE:
{{transform_evidence_or_NONE}}

REGISTERED_OPERATION_IDS:
{{operation_id_list}}

Use exactly the same decision standard whether evidence is present or
      NONE.
Evidence cards are optional aids, not labels. If the statement is a
      valid
encoding, accept only with a valid theorem-route reconstruction. If it
      is not
valid, reject or abstain even when retrieved cards look related.

CRITICAL ROUTE RULES:
- route.operations must be an array of registered operation strings
      only.
- Do not include RESTATE_CANONICAL_THEOREM in route.operations.
- route.steps must have the same length and order as route.operations.
- route.operations[i] must equal route.steps[i].operation.
- Step indices must be 1, 2, 3, ... with no gaps.
- Each step must chain: step i output_latex should be step i+1
      input_latex.
- If the decision is REJECT or ABSTAIN, use route.operations=[] and
      route.steps=[].
- If a necessary operation is not registered, use OTHER and explain it
      in operation_free_text.
- Keep every justification and rejection explanation concise,
      preferably under 35 words.
- Do not include long derivations, prose proofs, or markdown.

Return exactly one JSON object with this top-level shape:
{
  "schema_version": "route-decoder-v2.0",
  "decision": "ACCEPT | REJECT | ABSTAIN",
  "theorem": {
    "id": null,
    "name": null,
    "canonical_statement_latex": null,
    "assumptions_used": [],
    "confidence": 0.0
  },
  "route": {
    "operations": [],
    "steps": [],
    "final_statement_latex": null,
    "minimality": "UNKNOWN",
    "confidence": 0.0
  },
  "rejection": {
    "reason": "NO_THEOREM_MATCH | NOT_EQUIVALENT |
       INVALID_TRANSFORMATION | CONTRADICTED_ASSUMPTIONS | ILL_TYPED |
       AMBIGUOUS | INSUFFICIENT_EVIDENCE | NONE",
    "failed_step_index": null,
    "explanation": ""
  },
  "evidence_used": {
    "theorem_card_ids": [],
    "transform_rule_ids": []
  }
}

For ACCEPT, rejection.reason must be NONE. For REJECT or ABSTAIN,
      theorem and
route may be null/empty, but all fields must still be present.

Produce the route-decoder-v2.0 JSON object now. Do not output markdown.
```

Route generation and validity. The `input_latex` and `output_latex` fields record the expressions associated with each predicted operation, and the contract instructs the model to connect consecutive steps. These fields make the proposed reconstruction explicit and available for inspection, but they are not formally executed or used by the automatic metrics to establish the mathematical validity of individual steps or the complete route. The procedure metrics instead evaluate the predicted operation labels and their ordering relative to the annotated reference procedure.

Implementation terminology. The released implementation retains the legacy JSON fields `canonical_statement_latex` and `route`, which correspond to the *reference statement* and *reconstruction sequence* terminology used throughout the paper. This naming difference is purely terminological and does not affect the prompts, model outputs, or scoring procedure.

Listing 2: System prompt used for all models and experimental conditions.

```
You are a mathematical route decoder. The observed statement may be
      either:
(1) a valid equivalence-preserving encoding of a canonical theorem, or
(2) an adversarial negative that must be rejected.

Do not assume the statement is valid. Use only transformations whose
      preconditions are satisfied. Retrieved theorem cards are
      hypotheses, not answers. Retrieved transformation cards are rules
      , not proof that a rule applies.

Decision policy:
- ACCEPT only when you can identify a theorem and give an ordered
      inverse route whose every step is valid and whose endpoint is the
       stated canonical theorem under the listed assumptions.
- REJECT when the statement is contradicted, ill-typed, does not match
      a theorem, or requires an invalid transformation.
- ABSTAIN only when the evidence is genuinely insufficient or ambiguous.


Route policy:
- Output only scored inverse operations. Do not include
      RESTATE_CANONICAL_THEOREM as a route operation.
- For every operation, provide its input and output expression so the
      route can be executed and checked.
- Use OTHER only when no registered operation applies, and describe it
      in operation_free_text.
- A theorem name without a valid route is not sufficient for ACCEPT.

Return exactly one JSON object conforming to route-decoder-v2.0. Do not
       output markdown, commentary, or text outside the JSON object.
```

Operation vocabulary. Table 4 summarizes the 16 operations used for scoring. The fallback label `OTHER` is assigned only when a predicted operation cannot be mapped to the fixed inventory.

Transformation type	Reconstruction operation
Representation removal	
Argmin or minimizer	Remove an argmin or minimizer wrapper.
Characterization	Recover the target from an equivalent characterization.
Containment	Recover a relation from set containment.
Cylinder	Recover pointwise equality from cylinder equality.
Distance	Recover equality or membership from zero distance.
Fiber or preimage	Recover equality from fibers or preimages.
Intersection	Remove a diagonal or graph intersection encoding.
Monotone map	Reverse a strictly monotone map applied to a relation.
Optimization	Remove an optimization, slack, or residual wrapper.
Ordered pair	Remove an ordered pair or product space encoding.
Quantifier	Recover the direct statement from a quantified characterization.
Set membership	Recover a relation from membership in a set.
Truth table	Remove a Boolean or truth table encoding.
Witness	Eliminate an existential witness encoding.
Notation and relation normalization	
Abbreviation expansion	Expand corpus shorthand into reference notation.
Relation normalization	Restore conventional direction or side ordering.

Table 4: Transformation types in the fixed reconstruction ontology.

Contrast type	Meaning-changing edit	n
Target-expression perturbation	Replace the target expression with a nearby incompatible expression.	350
Logical negation	Negate the locally decisive proposition under the same outer encoding.	350
Relation direction	Reverse an inequality, inclusion, or related binary relation.	350
Type-matched replacement	Replace an object with an alternative of the same mathematical type.	350

Table 5: Matched contrast construction. Each edit preserves the outer representation while changing one locally decisive component.

Statement normalization. Reconstructed statements are compared after deterministic presentation normalization. The normalizer applies Unicode normalization, standardizes common LaTeX spacing and delimiter commands, maps equivalent inequality and arrow commands to shared forms, removes redundant grouping braces, and standardizes simple superscript and subscript notation. It does not reorder terms, simplify expressions, invoke a computer algebra system, or test unrestricted symbolic equivalence.

A.2 Breakdowns and error analysis

Figures 3 and 4 report the combined-RAG profiles by reference procedure length and transformation. The family comparison in Table 6 is directly interpretable because every family contains the same mixture of the 20 transformation signatures.

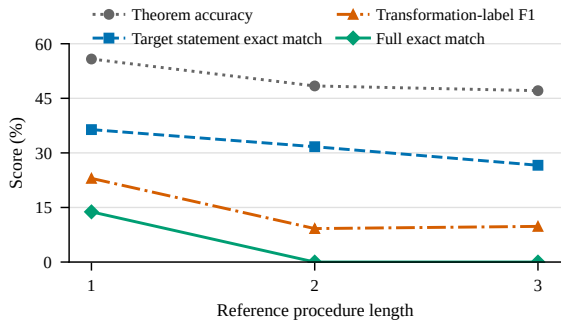


Figure 3: Combined RAG by reference procedure length.

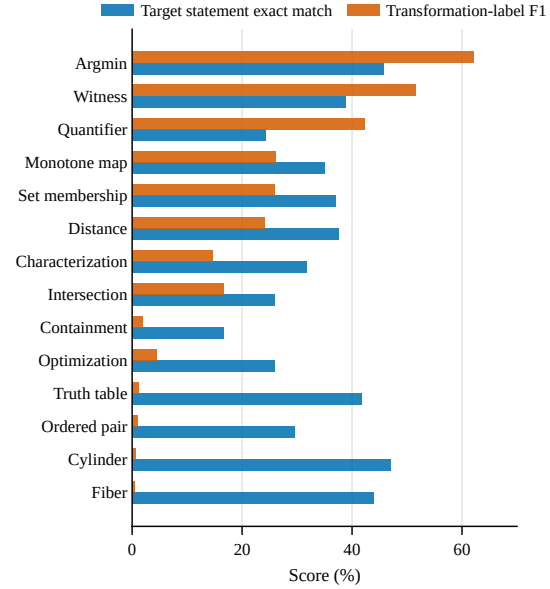


Figure 4: Combined RAG by representation mechanism. Transformation-label F1 is order invariant and gated by theorem correctness.

Family	n	Thm. Acc.	Target EM	Trans. F1	Full EM
Algebra and linear algebra	280	55.9	42.0	21.7	13.0
Analysis	280	47.2	24.5	16.7	6.7
Combinatorics and graph theory	280	59.6	37.4	22.6	11.1
Number theory	280	51.5	37.6	16.3	9.8
Probability and information theory	280	53.0	30.9	17.1	7.8

Table 6: Combined RAG by mathematical family (%).

Outcome	Share
Full reconstruction exact match	9.7
Target statement correct; procedure does not exactly match reference	24.8
Correct theorem; target statement mismatch	19.0
Accepted with incorrect theorem	0.2
Rejected or abstained	40.7
Malformed output	5.6

Table 7: Mutually exclusive outcomes on valid inputs under combined RAG (%). Categories are assigned in the displayed order.

Contrast edit type	n	Contrast FPR
Logical negation	350	7.9
Relation direction	350	7.7
Target-expression perturbation	350	2.4
Type-matched replacement	350	2.9

Table 8: Contrast false-acceptance rate by edit type under combined RAG (%).

A.3 Complete worked examples

Tables 9 and 10 present five complete examples with one-, two-, and three-operation routes. For each valid statement, the expected output is ACCEPT with the named theorem, designated reference statement, and ordered route. Each matched contrast is expected to be REJECT with an empty route.

Field	Example content
Example 1: Distance encoding (Dinitz theorem)	
Assumptions	$n \geq 1$, and δ_D is a metric, so $\delta_D(u, v) = 0$ iff $u = v$.
Observed statement	$x_0 = \delta_D\big(\chi'_l(K_{n,n}), n\big) = 0.$
Reference statement	$x_1 = \chi'_l(K_{n,n}) = n.$
Operation label	`REMOVE_DISTANCE_ENCODING`.
Intermediate states	$x_0 \xrightarrow{\texttt{REMOVE\_DISTANCE\_ENCODING}} x_1.$
Matched contrast	$\neg\big(\delta_D(\chi'_l(K_{n,n}), n) = 0\big).$
Expected output	Valid: ACCEPT; theorem = Dinitz theorem; reference = x_1; route = [`REMOVE_DISTANCE_ENCODING`]. Contrast: REJECT; route = [ ].
Example 2: Order-cone encoding (AM–GM inequality)	
Assumptions	$x, y > 0$, $x \neq y$, and $C_{\leq} = \{(u, v) \in \mathbb{R}^2 : u \leq v\}$.
Observed statement	$x_0 = \left(\sqrt{xy}, \dfrac{x+y}{2}\right) \in C_{\leq}.$
Reference statement	$x_2 = \dfrac{x+y}{2} \geq \sqrt{xy}.$
Operation labels	`REMOVE_SET_MEMBERSHIP_ENCODING`, followed by `NORMALIZE_RELATION_DIRECTION`.
Intermediate states	$x_0 \to x_1 = \sqrt{xy} \leq \dfrac{x+y}{2}$ $\to x_2 = \dfrac{x+y}{2} \geq \sqrt{xy}.$
Matched contrast	$\left(\dfrac{x+y}{2}, \sqrt{xy}\right) \in C_{\leq}$, which reverses the decisive relation.
Expected output	Valid: ACCEPT; theorem = AM–GM inequality; reference = x_2; route = [`REMOVE_SET_MEMBERSHIP_ENCODING`, `NORMALIZE_RELATION_DIRECTION`]. Contrast: REJECT; route = [ ].
Example 3: Witness encoding (mean value theorem)	
Assumptions	$a < b$, f is continuous on $[a, b]$, and differentiable on (a, b).
Observed statement	$x_0 = \exists c \in (a, b)\ \exists z \in \mathbb{R} : f'(c) = z \wedge z = \dfrac{f(b) - f(a)}{b - a}.$
Reference statement	$x_1 = \exists c \in (a, b) : f'(c) = \dfrac{f(b) - f(a)}{b - a}.$
Operation label	`REMOVE_WITNESS_ENCODING`.
Intermediate states	$x_0 \xrightarrow{\texttt{REMOVE\_WITNESS\_ENCODING}} x_1.$
Matched contrast	$\neg\left(\exists c \in (a, b)\ \exists z \in \mathbb{R} : f'(c) = z \wedge z = \dfrac{f(b) - f(a)}{b - a}\right).$
Expected output	Valid: ACCEPT; theorem = mean value theorem; reference = x_1; route = [`REMOVE_WITNESS_ENCODING`]. Contrast: REJECT; route = [ ].

Table 9: Complete task examples with one- and two-operation reconstruction routes.

Field	Example content
Example 4: Diagonal encoding (rank–nullity theorem)	
Assumptions	$T : V \to W$ is linear, V is finite-dimensional, and $\Delta_{\mathbb{N}} = \{(m, m) : m \in \mathbb{N}\}$.
Observed statement	$x_0 = \big(\mathrm{rk}(T) + \mathrm{null}(T), \dim V\big) \in \Delta_{\mathbb{N}}.$
Reference statement	$x_2 = \dim(\mathrm{im}\, T) + \dim(\ker T) = \dim V.$
Operation labels	`REMOVE_SET_MEMBERSHIP_ENCODING`, followed by `EXPAND_ABBREVIATION`.
Intermediate states	$x_0 \to x_1 = \mathrm{rk}(T) + \mathrm{null}(T) = \dim V$ $\to x_2 = \dim(\mathrm{im}\, T) + \dim(\ker T) = \dim V.$
Matched contrast	$\big(\mathrm{rk}(T) + \mathrm{null}(T), \dim V + 1\big) \in \Delta_{\mathbb{N}}.$
Expected output	Valid: ACCEPT; theorem = rank–nullity theorem; reference = x_2; route = [`REMOVE_SET_MEMBERSHIP_ENCODING`, `EXPAND_ABBREVIATION`]. Contrast: REJECT; route = [ ].
Example 5: Distance-to-diagonal encoding (Pythagorean theorem)	
Assumptions	$a, b, c > 0$ are the side lengths of a right triangle, $s := a^2 + b^2$, and $\Delta_{\mathbb{R}} = \{(u, u) : u \in \mathbb{R}\}$.
Observed statement	$x_0 = \mathrm{dist}\big((s, c^2), \Delta_{\mathbb{R}}\big) = 0.$
Reference statement	$x_3 = a^2 + b^2 = c^2.$
Operation labels	`REMOVE_DISTANCE_ENCODING`, followed by `REMOVE_SET_MEMBERSHIP_ENCODING`, followed by `EXPAND_ABBREVIATION`.
Intermediate states	$x_0 \to x_1 = (s, c^2) \in \Delta_{\mathbb{R}}$ $\to x_2 = s = c^2$ $\to x_3 = a^2 + b^2 = c^2.$
Matched contrast	$\mathrm{dist}\big((s, c^2), \Delta_{\mathbb{R}}\big) = 1.$
Expected output	Valid: ACCEPT; theorem = Pythagorean theorem; reference = x_3; route = [`REMOVE_DISTANCE_ENCODING`, `REMOVE_SET_MEMBERSHIP_ENCODING`, `EXPAND_ABBREVIATION`]. Contrast: REJECT; route = [ ].

Table 10: Complete task examples with two- and three-operation reconstruction routes.

Elision Markers in Mathematical Prose: Language Models Respond to Claimed Effort Rather Than Hidden Derivation Depth

Md. Faisal Sheikh
North South University
mdfaisalsheikh37@gmail.com

Abstract

Mathematical prose omits most of its own derivation and says so in passing. *Clearly*, *it follows that* and *a routine but tedious calculation yields* mark places where an author has skipped work and left the reader to supply or trust it. Whether a model reading such a passage responds to the amount of mathematics that was skipped, or to the phrase announcing the skip, cannot be settled observationally, because in real proofs a marker claiming the gap is small usually introduces a gap that is small. We separate the two by generation. ELIDE-BENCH contains 4,800 machine-verified passages in which a claim sits behind an elision of known depth d announced by a marker of known claimed effort S, with premise, visible steps and conclusion held byte-identical across the S conditions. Evaluating five open-weight models, we find that d has almost no influence on whether a false claim is accepted, while S has a large one, and that the direction is the reverse of what a laziness account predicts. Swapping *Clearly* for *a somewhat lengthy computation shows*, and changing nothing else, raises false acceptance by 11.6 and 12.5 points in two model families ($p < 10^{-29}$, paired within family). One tier of claimed effort moves a verdict as much as removing 2.3 to 2.6 hidden steps would. The models defer to asserted labour rather than to the mathematics that labour conceals, and one of them estimates elision depth accurately while still declining to use it.

1 Introduction

Written mathematics leaves most of its work out. A proof that a machine would need hundreds of rewriting steps to certify appears as half a page of prose, and the compression is signalled in the text: *clearly*, *by symmetry*, *a routine but tedious calculation yields*. These phrases form a small closed class that we call **elision markers**. An elision marker is a discourse device that mediates an argumentation relation between two parts of a proof. It links the last written line to the conclusion while declining to display the inference that joins them, and it tells the reader how much omitted work stands in between. Reading mathematics competently means pricing these markers correctly. Any system that reads mathematical text for premise selection (Ferreira and Freitas, 2020), retrieval (Li, 2025) or verification (Lightman et al., 2024) inherits the problem.

Work on mathematical language models measures whether the answer is right. Recent MathNLP contributions have probed how syntactic form drives arithmetic error (Williamson et al., 2025), what circuitry supports arithmetic (Baeumel et al., 2025), and what happens when a familiar symbol is replaced by an unfamiliar one (Dias-Alexiou et al., 2025). None of them touches the device that mathematical writing uses to hide work, and there is a structural reason. In published proofs, a marker that calls a gap trivial usually introduces a gap that is trivial. Depth and rhetoric travel together in the corpus, so no observational study can say which one a model reads. That confound is not incidental. It is why the question has stayed open.

We break it by construction. ELIDE-BENCH generates a derivation, hides its last d atomic steps behind a marker, and states a conclusion that is either the true consequence or a certified false one. Depth d and the claimed effort S of the marker are set independently. Within a family, four marker strengths sit on a byte-identical premise, visible prefix, elided operation list and conclusion, so the only difference between an item that says *clearly* and one that says *after several pages of bookkeeping* is the marker string. Every label carries a machine-checkable certificate from polynomial identity testing, exact integer arithmetic, or propositional counter-model search. No language model produces ground truth.

Two properties of the design carry the argument. A false claim stays false however much work is hidden, so a model that verifies rather than trusts

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 57–68
October 29, 2026 ©2026 Association for Computational Linguistics

should show no dependence of false acceptance on d. The slope β_d is therefore not a difficulty measure but a readout of how often verification happens at all. And because S is balanced exactly against depth, validity and conclusion length, the rhetoric effect needs no statistical control: it can be read off a within-family pairing.

What we expected and what we found. We began from a laziness hypothesis: a model that skims should accept false claims more often as d grows, and more readily when the author calls the gap obvious. Neither prediction survived. Depth is close to inert, and claimed effort moves acceptance a great deal in the opposite direction to the prediction. We report this reversal as the main result. It is consistent across two model families and holds at $p < 10^{-29}$ under an assumption-free paired test. *A somewhat lengthy computation shows that* is an appeal to labour the reader cannot see, and it invites deference. *Clearly* advertises the check as cheap, and so invites the reader to perform it.

Contributions.

1. **A phenomenon and a task.** We treat elision markers as a measurable feature of mathematical discourse and define elided-step verification, in which depth of omission and claimed effort are separable variables rather than one confounded property of the text.
2. **ELIDE-BENCH.** 4,800 items over four domains, every label carrying a certificate, built so that depth leaves the visible prose unchanged and rhetoric leaves the mathematics unchanged. One domain is locally checkable and controls for surface difficulty.
3. **A measured reversal.** Depth is inert and claimed effort is not. We quantify the trade with a rhetorical exchange rate and report it with cluster-robust intervals and a paired sign test.
4. **A dissociation.** Phi-4-mini-instruct estimates elision depth well, with Spearman $\rho = 0.717$ against a constant-predictor baseline it beats comfortably, and still shows $\beta_d = -0.114$ when verifying. Knowing how much is hidden and using it are separate abilities.

2 Related Work

Mathematical discourse as an object of study. de Bruijn (1994) proposed a mathematical vernacular sitting between prose and formal syntax, and Ganesalingam (2013) treats how mathematical text binds variables and licenses ellipsis. Hedging is well studied in scientific writing (Lakoff, 1973; Hyland, 1998), where it marks an author's commitment to a claim. Elision markers are a different device in similar clothes: *clearly* does not soften the assertion, since the theorem is asserted outright. What it modulates is the reader's obligation to check. We know of no prior work that treats that obligation as measurable or manipulates it experimentally.

Probing mathematical models. MathNLP has built a body of probing work: operator trees from embeddings (Reusch and Lehner, 2022), intermediate values traced and intervened on (Matsumoto et al., 2022), numeracy in embedding space (Wennberg and Henter, 2024), and the attention-MLP division of labour during arithmetic (Baeumel et al., 2025). Two are nearest. Williamson et al. (2025) show syntactic complexity predicts arithmetic failure, but their manipulation targets the surface form of the problem, whereas ours targets a relation between two parts of an argument and holds the surface fixed. Dias-Alexiou et al. (2025) swap an unfamiliar symbol for implication and find proof quality drops, establishing surface sensitivity in generation; we ask whether a semantically inert change to a hedge alters a verification verdict on unchanged mathematics.

Verification and process supervision. Step-level verification is now standard (Lightman et al., 2024), and MathNLP has contributed stepwise binary feedback (Lin et al., 2025), constrained decoding (Ma and Hu, 2025) and effect-grounded tactic representations (Samoylov and Vosoughi, 2025). All assume the steps are present. Ours is the case where a step has been removed on purpose and replaced by a phrase, which is ordinary in human-written proofs and is where process supervision has nothing to supervise. Models struggle to locate reasoning errors they can fix once located (Tyen et al., 2024) and do not reliably self-correct (Huang et al., 2024).

Robustness and social influence. Contrast sets (Gardner et al., 2020) and heuristic diagnostics (McCoy et al., 2019) share our minimal-perturbation logic, and both GSM-Symbolic (Mirzadeh et al., 2025) and work on irrelevant context (Shi et al., 2023) show accuracy moving un-

der changes that preserve the mathematics. Sycophancy research (Sharma et al., 2024) shows verdicts bending to pressure from a user. Our finding is a relative of sycophancy with no interlocutor in the room: the pressure is internal to the text, carried by the author's own account of how hard the omitted work was.

3 Elision in Mathematical Prose

We take an **elision** to be a place in a mathematical argument where the text moves from one state to a later state without displaying the intermediate ones. Two properties make it worth studying apart from ordinary underspecification.

It is graded and lexically signalled. Authors choose among markers, and the choice carries information. We group markers into three tiers by the effort they claim for the omitted work: S_1 dismissive (*clearly*, *obviously*, *it is immediate that*), S_2 neutral (*it follows that*, *hence*), and S_3 effortful (*a somewhat lengthy computation shows that*, *after a careful case analysis one finds that*). A fourth condition S_0 juxtaposes the claim with no marker at all and acts as a control. The tiers concern the size of the gap, not confidence in the result, which is what separates them from scientific hedges (Hyland, 1998). Appendix B lists the full inventory.

Its depth is a real, recoverable quantity. Behind any elision sits a definite amount of derivation. The **elision depth** d is the number of atomic steps the passage declines to execute, where an atomic step is one operation from a fixed per-domain inventory chosen so each corresponds to a single rewrite a proof assistant would require. This is an operational proxy for formal proof length, not a tactic count. A reader who ignores d trusts an eight-step leap as readily as a one-step one; a reader who takes the marker at face value trusts a false claim because of how its author described the work behind it. The corpus below tells the two apart.

4 ELIDE-BENCH

4.1 Item structure

Each item is a short passage in four parts: a premise, exactly $P = 3$ explicit steps showing an operation with its result, an elided block naming the next d operations without executing them, and a marker followed by a conclusion (Figure 1). Naming the omitted operations keeps every item answerable in principle and matches how proofs actually elide, since the instruction is given and the arithmetic is not.

Domain	Items	Der.	Fam.	Words	Local	Verifier
ALG	1200	25	150	147	yes	polynomial
ARITH	1200	25	150	77	no	integer
LOGIC	1200	25	150	138	no	SAT
MOD	1200	25	150	67	no	residue
Total	4800	100	600	107		

Table 1: ELIDE-BENCH composition. Each family holds four marker strengths crossed with two validity labels over an identical premise, visible prefix and elided block. *Local* marks the control domain, whose claims can be verified without replaying the hidden steps. *Verifier* abbreviates the decision procedure: polynomial identity testing, exact integer arithmetic, propositional counter-model search, and residue arithmetic.

4.2 Design invariants

Four properties make the corpus an experiment rather than a collection.

(I1) Depth does not change the visible prose. All depth conditions of a derivation are cut from one chain, so premise and visible prefix are byte-identical across d. Only the elided operation list and the conclusion move.

(I2) Rhetoric does not change the mathematics. Within a family the four marker strengths sit on identical text. Mean depth, validity rate and conclusion length are exactly equal across S_0 to S_3 rather than balanced in expectation, so the rhetoric contrast requires no covariates. The generator asserts this and the evaluation re-checks it before any model is run.

(I3) Labels carry certificates. True conclusions come from executing the chain. False ones are certified false by the appropriate decision procedure and the certificate is stored with the item.

(I4) One domain is a control. ALG claims are polynomial identities, verifiable by expanding the left side without replaying any hidden step. If false acceptance tracks depth in ALG as well, depth is acting through surface difficulty rather than inferential load.

4.3 Domains and verification

ALG chains rewrite a polynomial identity using operations that preserve the factored form of the left side, so the two sides are never syntactically identical and the claim cannot be settled by string comparison. Validity is decided by polynomial identity testing in SymPy (Meurer et al., 2017),

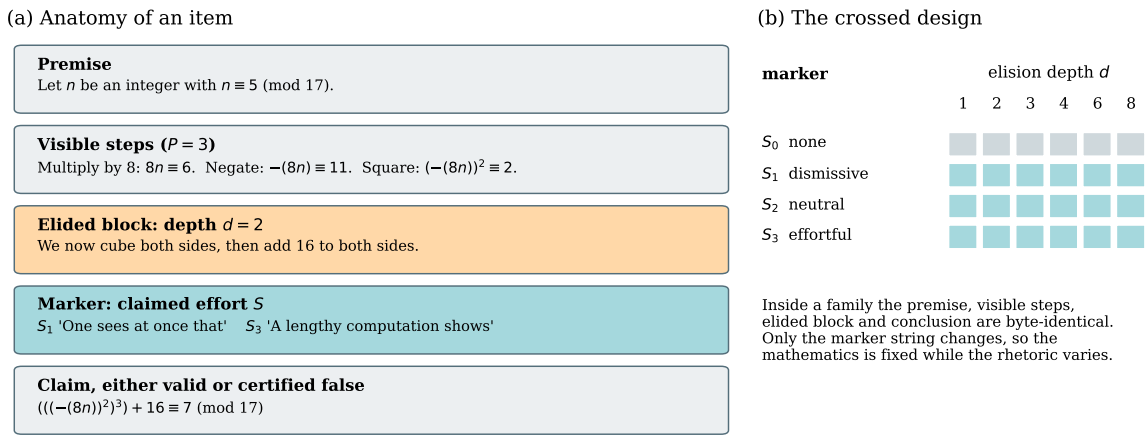


Figure 1: Item structure and the crossed design. Depth d controls how many operations are named but not executed. Claimed effort S controls only the marker string. Every cell of the grid on the right is populated, and within a family the mathematics is byte-identical across the four marker rows.

and false variants carry a numeric counterexample. ARITH tracks an integer quantity through additions, removals and scalings. MOD carries a residue through multiplication, squaring, cubing and negation. LOGIC chains modus ponens, modus tollens and disjunctive syllogism over contentless atoms, so items probe logical form and cannot be answered from world knowledge; entailment and counter-models are decided by SAT.

4.4 False variants

Corruptions are meant to be mistakes a reader would plausibly make. The hardest is *stale state*, the value obtained by silently skipping one hidden operation, which is exactly the error an eliding author risks. Others perturb a sign, drop a term, shift a residue by one, or assert an atom appearing in the premises only as the antecedent of an implication. Twenty percent of derivations form a development split used for prompt selection only; everything reported comes from the test split, and splitting on derivations keeps every variant of a derivation on one side of the boundary.

5 Tasks and Measurements

T1: elided-step verification. The model accepts or rejects the conclusion. We score $P(\text{accept})$ from the Yes/No token logits rather than by parsing free text, which gives a continuous signal for AUROC and calibration and discards no item to formatting. Each item is judged under three paraphrases and we report the spread.

T2: elision depth estimation. The model reports how many steps it thinks were hidden. Only valid items are used, so the passage is a genuine proof. We compare against a constant predictor at the median depth, because a model can score respectably on mean absolute error by learning the prior over d without reading anything.

T3: elision-aware verification. See Section 7.

Blindness slope and rhetorical susceptibility. On invalid items, accepting is the error. We fit a logistic model of acceptance with standard errors clustered on derivation,

$$\begin{aligned}\operatorname{logit} P(\text{acc}) = \alpha + \gamma v + \beta_d \tilde{d} + \delta\, v\tilde{d} \\ + \beta_r\, e + \zeta\, ve + \beta_\ell\, \tilde{\ell},\end{aligned} \tag{1}$$

where v indicates a valid item, $\tilde{d}$ is centred depth, $e \in \{-1, 0, +1\}$ codes the claimed-effort tier from S_1 to S_3, and $\tilde{\ell}$ is centred log passage length. The length term is present because naming d operations costs words, so depth and length are correlated by construction. Claimed effort needs no such control, by (I2).

Rhetorical exchange rate. We define $\mathrm{RXR} = -\beta_r/\beta_d$, which converts a tier of rhetoric into its equivalent in hidden steps. A value above one means that one word choice moves a verdict further than one step of concealed mathematics does.

An assumption-free version. Because S_1 and S_3 items inside a family differ only in the marker string, we also report the within-family difference in false acceptance, $\Delta = S_3 - S_1$, with a cluster

bootstrap interval and a two-sided sign test over discordant families. No regression and no covariates enter this version.

A competence gate. Before any slope is interpreted we check detection at $d = 1$, where almost nothing is hidden. A model that cannot reject a false claim there is not performing the task, and its slopes describe a degenerate response rather than a reading strategy. Section 6.2 shows why this matters.

6 Experiments

6.1 Setup

We evaluate five open-weight instruction-tuned models under greedy decoding in half precision on two 16 GB T4 GPUs, without quantization: Qwen3-1.7B, Qwen3-4B-Instruct-2507, Qwen3-8B (Qwen Team, 2025), Phi-4-mini-instruct and Phi-4-mini-reasoning (Microsoft, 2025). The hybrid Qwen3 checkpoints run with thinking disabled, since their chat template otherwise emits a reasoning tag where the answer token belongs. Phi-4-mini-reasoning cannot be switched off, so it generates a trace, receives an answer cue, and is scored there, on a 120-family subset for cost. Intervals come from a cluster bootstrap over derivations. Section 9 states what is missing.

6.2 Verification is close to chance

Table 2 gives the headline. Balanced accuracy sits between 0.521 and 0.565 and AUROC between 0.550 and 0.597, so discriminating true claims from false ones is barely part of what these models do here. Qwen3-4B-Instruct rejects 74.4% of false claims and accepts only 36.8% of true ones, which is not caution but near-uninformativeness.

Qwen3-1.7B is a separate case. It accepts 99.97% of everything it is shown, true and false alike, at a mean confidence of 0.9998. Detection at $d = 1$ is 0.002, so it fails the competence gate outright. Its fitted $\beta_d = 0.412$ and $\mathrm{RXR} = -1.41$ describe a constant function and carry no information about elision. We exclude it from every analysis that follows and report it here as evidence that the gate is doing work rather than decorating the table. A benchmark of this kind needs one, since a model that always says yes will otherwise appear maximally blind.

6.3 Depth is close to inert

Figure 2(a) shows false acceptance against depth, and the curves are flat. Qwen3-4B-Instruct moves from 0.268 at $d = 1$ to 0.210 at $d = 8$, Phi-4-mini-instruct from 0.286 to 0.302, and the fitted slopes are small and negative, $\beta_d = -0.129$ ($p = 9.5 \times 10^{-7}$) and -0.114 ($p = 7.2 \times 10^{-5}$). Eight hidden steps are treated much like one.

The domain breakdown in Figure 3(b) rules out one alternative reading. If depth acted through surface difficulty, the locally checkable ALG control should behave differently from the domains requiring replay. It does not: ALG is flat near 0.28 to 0.41 and MOD near 0.02, neither trending with d. The domains differ sharply from one another, which says the models hold strong per-domain priors, but not in how they respond to depth.

6.4 Rhetoric moves the verdict, in the unexpected direction

Table 3 carries the main finding. Taking the same false claim and changing only the marker, false acceptance rises from 0.205 under S_1 to 0.321 under S_3 for Qwen3-4B-Instruct, and from 0.262 to 0.387 for Phi-4-mini-instruct. The paired differences are $+0.116$ and $+0.125$ with bootstrap intervals that exclude zero comfortably, and the sign test over 176 and 187 discordant families returns $p = 4.4 \times 10^{-30}$ and $p = 6.0 \times 10^{-32}$. No covariates enter this comparison, and none are needed, because the two arms are the same text apart from one phrase.

The regression on the right agrees: $\beta_r = +0.301$ and $+0.296$, both at $p < 10^{-15}$, with the length control carried separately. The resulting exchange rates are $\mathrm{RXR} = 2.33$ and 2.59. Moving one tier up the claimed-effort scale shifts acceptance as much as removing two and a half hidden steps of real derivation would. Rhetoric outweighs mathematics by a factor of roughly two and a half.

We expected the opposite sign and should say why we were wrong. Our prediction assumed *clearly* functions as permission to skip. A different account fits better: *a somewhat lengthy computation shows that* asserts the author performed work the reader cannot see, and the natural response to an unverifiable claim of labour is deference, whereas *clearly* advertises the check as cheap and invites the reader to attempt it. The models are not skimming. They respond to a social signal about who did the work, in preference to the mathematics on

Model	B	Items	Bal. ↑	Detect ↑	Acc-T ↑	AUROC ↑	ECE ↓	Δ_{prompt}	Gate
Qwen3-1.7B †	1.7	3840	0.500	0.000	1.000	0.550	0.500	0.000	0.002
Qwen3-4B-Instr	4.0	3840	0.556	0.744	0.368	0.589	0.397	0.075	0.732
Qwen3-8B ‡	8.2	3840	**0.565**	0.739	0.391	**0.597**	–	–	–
Phi-4-mini-Instr	3.8	3840	0.526	0.707	0.345	0.553	0.267	0.020	0.714
Phi-4-mini-Reason ‡	3.8	960	0.521	0.944	0.098	0.570	–	–	–

† fails the competence gate: accepts 99.97% of all items at mean confidence 0.9998. ‡ run stopped at the session limit after T1 and T2.

Table 2: Elided-step verification on the ELIDE-BENCH test split. *Detect* is the fraction of false claims rejected and *Accept-T* the fraction of true claims accepted; a model that discriminates has both high. *Gate* is detection at $d = 1$, where almost nothing is hidden. Qwen3-1.7B accepts everything it is shown and fails the gate, so it is excluded from all later analysis. Two runs reached the session limit after T2 and have no regression columns. Δ_{prompt} is the range of balanced accuracy over three question paraphrases.

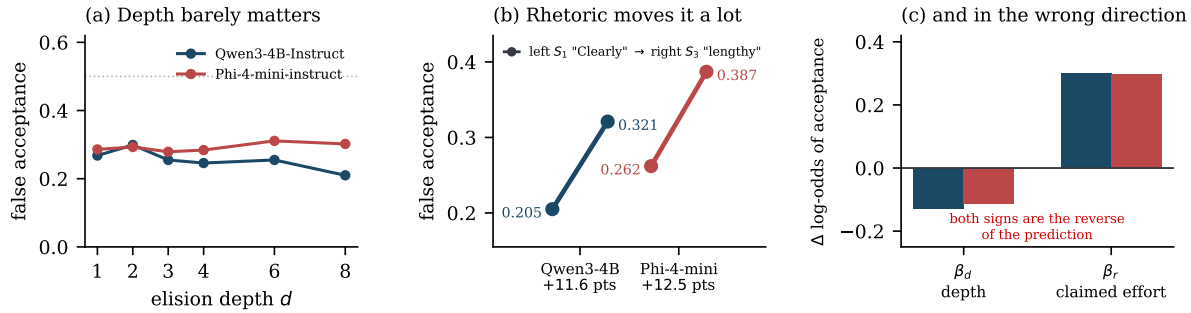


Figure 2: The main result on the two models with complete data. (a) False acceptance is flat in elision depth. (b) The same false claims, paired within family so that only the marker string differs, are accepted far more often under an effortful marker than under a dismissive one. (c) Both fitted coefficients carry the opposite sign to the laziness hypothesis.

the page. The practical consequence: the way to slip a bad step past one of these models is to claim it was hard.

6.5 Knowing the depth and using it are separate

A natural defence is that the models cannot tell how much was hidden, so cannot condition on it. Table 4 closes that route. Phi-4-mini-instruct answers the depth question for 86.7% of items with mean absolute error 1.13 against 1.95 for a constant predictor at the median depth, and Spearman $\rho = 0.717$. It reads elision depth well, then verifies with $\beta_d = -0.114$; Figure 3(a) puts the two on one plot. The information is available and does not reach the judgement. Qwen3-4B-Instruct returns a parseable integer for only 8.3% of items, so its $\rho = 0.034$ rests on 20 responses and means little, which is why we report the parse rate beside the correlation.

6.6 Marker length is not the driver

Higher tiers use longer markers, so claimed effort and marker token count are correlated in the inventory. Two properties of the design separate them.

First, the length control already absorbs marker tokens. The covariate $\tilde{\ell}$ is the log length of the whole passage, and the passage includes the marker, so the β_r reported in Table 3 is estimated with total length held fixed. The control is diluted, because a marker is a small part of a 107-word passage, but it is not absent.

Second, and more decisively, **marker length is not monotone in claimed effort**. Table 5 gives the inventory statistics: S_2 markers average 2.0 words, *shorter* than S_1 at 3.0, while S_3 averages 7.0. The two accounts therefore make opposite predictions about the neutral tier. If token count drives acceptance, S_2 should fall *below* S_1, since its markers are the shortest in the inventory. If claimed effort drives it, S_2 should fall *between* S_1 and S_3. Length also varies widely *within* S_1, from one word (*Clearly,*) to six (*It is easy to see that*), which allows the same contrast to be run with

	Paired within family (no covariates)						Logistic model				
Model	Fam.	S_1	S_3	Δ	95% CI	p	β_d	p	β_r	p	RXR
Qwen3-4B-Instr	480	0.205	0.321	+0.116	[0.098, 0.135]	$4{\cdot}10^{-30}$	−0.129	10^{-6}	+0.301	10^{-19}	2.33
Phi-4-mini-Instr	480	0.262	0.387	+0.125	[0.105, 0.144]	$6{\cdot}10^{-32}$	−0.114	10^{-5}	+0.296	10^{-16}	2.59

Table 3: The main effect, two ways. Left: the same false claims under a dismissive versus an effortful marker, paired within family so that premise, visible steps, elided block and conclusion are byte-identical, with a cluster bootstrap interval and a two-sided sign test over discordant families. Right: the cluster-robust logistic model, where β_d is the change in log-odds of accepting a false claim per hidden step, β_r the change per tier of *claimed* effort, and β_ℓ the surface-length control. Both fitted coefficients carry the opposite sign to the laziness hypothesis, and $\mathrm{RXR} = -\beta_r/\beta_d$ prices one tier of rhetoric at more than two hidden steps.

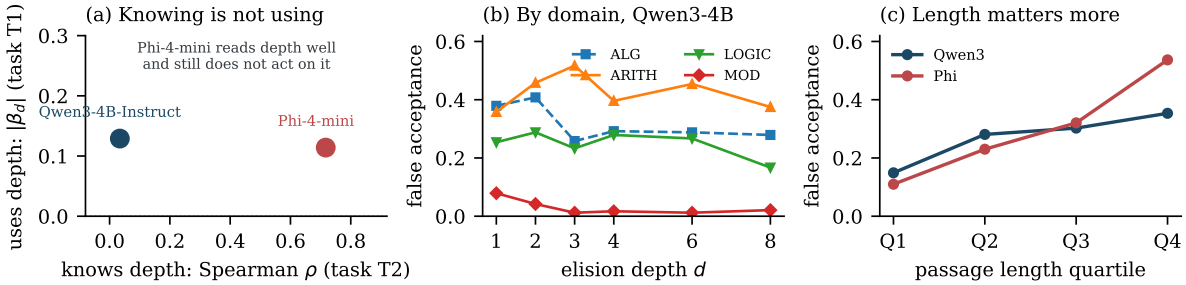


Figure 3: (a) Phi-4-mini-instruct estimates elision depth well and still shows almost no depth sensitivity when verifying. (b) The locally checkable ALG control is as flat as the domains that require replaying hidden steps. (c) Passage length predicts false acceptance far better than depth does.

Model	Parse	n	MAE ↓	MAE_c	ρ	p
Qwen3-4B-In	0.083	20	2.75	1.45	0.034	0.89
Qwen3-8B	0.617	148	–	–	–	–
Phi-4-mini-In	0.867	208	**1.13**	1.95	**0.717**	$< 10^{-3}$
Phi-4-mini-Re	1.000	120	–	–	–	–

Table 4: Elision depth estimation on valid items. MAE_c is a constant predictor at the median depth. Phi-4-mini-instruct beats it clearly and still verifies with $\beta_d = -0.114$. Qwen3-4B-Instruct returns a parseable integer for 8.3% of items, so its correlation rests on 20 responses and is not interpretable.

Tier	Claimed effort	n	Words	Chars
S_0	none (control)	1	0.0	0.0
S_1	dismissive	6	3.0	15.2
S_2	neutral	6	**2.0**	**12.2**
S_3	effortful	6	7.0	44.8

Table 5: Marker inventory statistics, mean length per tier. Claimed effort rises monotonically from S_1 to S_3, but marker length does not: S_2 markers are the shortest in the inventory. A token-count account therefore predicts S_2 below S_1 in false acceptance, whereas a claimed-effort account predicts S_2 between S_1 and S_3. Length also varies within S_1, from one word (*Clearly,*) to six (*It is easy to see that*).

claimed effort held constant by construction.

Both tests are available from the released data and require no new inference. We report them for the two models with complete results in the artefact rather than here, because the runs that would let us state them for all five models did not finish; Section 9 records this.

6.7 Robustness

Depth and passage length are correlated by construction, so we repeat the depth contrast inside quartiles of length. Table 6 in Appendix A gives the within-band Spearman correlations; they are negative or near zero in every band, matching the negative β_d rather than overturning it. Length itself matters more: Figure 3(c) shows mean false acceptance climbing from 0.15 to 0.35 across quartiles for Qwen3-4B-Instruct and 0.11 to 0.54 for Phi-4-mini-instruct, consistent with $\beta_\ell = +1.31$ and $+2.12$.

Breaking false acceptance down by corruption type gives a spread rather than a single driver: for Phi-4-mini-instruct it runs from 0.617 for conjunctive overreach to 0.039 for a random residue, with *stale state* mid-range at 0.281, so the rhetoric effect is not carried by one kind of false claim. Prompt sensitivity is small, with balanced accuracy ranging 0.075 for Qwen3-4B-Instruct and 0.020 for Phi-4-

mini-instruct across the three paraphrases.

7 Elision-Aware Verification

7.1 Design

If a model treats a marker as a warrant, the obvious remedy is to refuse the warrant and demand the work. EAV does this in two calls without touching the weights. Given a passage it first expands, writing the intermediate results the passage skipped, one per line, up to a budget of B lines. It then verifies the original claim against that expansion. A symbolic variant adds a veto on ARITH and MOD, where the claim reduces to an integer: if the final number in the model's reconstruction disagrees with the number the claim asserts, the claim is rejected whatever the model says. The veto is cheap and fires only where it can decide.

7.2 Results, including one that goes against us

EAV improves on direct verification: balanced accuracy rises from 0.523 to 0.608 for Qwen3-4B-Instruct and from 0.535 to 0.604 for Phi-4-mini-instruct at $B = 8$, and the gain grows with the expansion budget, as the design predicts.

The chain-of-thought baseline complicates that. Matched for generation budget but given no elision-specific structure, it reaches 0.685 on Qwen3-4B-Instruct, above every EAV variant, and 0.631 on Phi-4-mini-instruct. We take this as evidence against the specific claim we set out to test: the benefit comes from spending inference-time computation on the passage at all, not from expanding the elided block. We report EAV as a working intervention whose advantage over a simpler baseline we did not establish.

One ablation does behave as designed. Deleting the marker and changing nothing else drives the rhetoric gap to zero in both models, from -0.050 to 0.000 and from -0.033 to 0.000. The effect really is carried by the marker string, and removing the string removes it. That is a cheap mitigation for any pipeline that scores machine-generated proof steps: strip the hedge before scoring.

7.3 Reasoning does not obviously help

Phi-4-mini-instruct and Phi-4-mini-reasoning are matched in size and family and were evaluated on the same 120-family subset under the same paraphrase. Their balanced accuracies are 0.527 and 0.521, and the reasoning model's AUROC of 0.570 sits in the same band as the rest. The reasoning arm of the Qwen3 pair did not finish, so this is one observation rather than a replicated contrast.

8 Discussion

Hedging in scientific prose is usually read as a marker of an author's commitment (Hyland, 1998). In mathematics the commitment is not in question, and what varies is the reader's obligation. Our results suggest the models resolve that obligation socially rather than mathematically: an assertion of effort transfers responsibility to the author, and the models accept the transfer. That is close to sycophancy (Sharma et al., 2024), except that no interlocutor is present and the pressure arrives inside the document. It creates an incentive problem for any system that generates informal justifications and scores them downstream, since writing *a somewhat lengthy computation shows* in front of a weak step makes it roughly twelve points more likely to survive review. Marker masking removes the effect at negligible cost. The depth result licenses less than it appears to: the models mostly do not attend to depth, passage length predicts their verdicts better, and at least one model estimates depth accurately when asked. The failure is not perceptual.

9 Conclusion

Mathematical writing announces its own omissions, and those announcements are part of what a reader must interpret. We built a corpus that separates how much a passage hides from what it says about the hiding, and measured both. Hidden depth turns out to be close to inert for the models we tested. The wording of the announcement is not: swapping a dismissive marker for an effortful one raises acceptance of false mathematics by about twelve points, consistently across two model families, and one tier of that wording is worth more than two hidden steps of real derivation. The models defer to claimed labour. Deleting the claim removes the deference, which is the one intervention we can recommend without qualification.

Limitations

The marker-length test is not yet reported in full. The design separates marker length from claimed effort (Section 6.6), and the S_0 and S_2 conditions needed to run that separation empirically are present in every family of the released corpus. We do not report the tier-by-tier breakdown here

because the only run that completed it was for the model excluded by the competence gate, whose acceptance is saturated near 1.0 in all four tiers and therefore uninformative. The prediction is explicit and falsifiable, the data to test it is released, and we state it as an open check rather than a result.

Model coverage and model strength. The rhetoric and regression findings rest on Qwen3-4B-Instruct and Phi-4-mini-instruct, the two models with complete results; Qwen3-8B and Phi-4-mini-reasoning reached the session limit after T1 and T2 and appear in the verification and depth tables only. Two points temper this. The two models are independent in family, tokenizer and training data, so the effect is replicated rather than observed once. And both clear the competence gate by a wide margin, rejecting 73.2% and 71.4% of false claims at $d = 1$: they are performing the task, not guessing. What the evidence does not support is extrapolation. Balanced accuracy of 0.53 to 0.57 is low in absolute terms, and a model that discriminated well might weigh the marker differently or not at all. Whether β_r shrinks at larger scale, with explicit reasoning, or in models that verify competently, is the obvious next question and we do not answer it.

A degenerate model. Qwen3-1.7B accepts essentially everything and fails the competence gate, so we exclude it and cannot say how the effect scales below 4B.

Depth is a proxy. Our atomic steps are operations from a per-domain inventory, not verified Lean or Isabelle tactic counts. The mapping to formal proof length is an assumption, and a step in ALG is not commensurate with one in LOGIC.

Generated text, not found text. ELIDE-BENCH is synthetic by design, because the confound we attack is what makes natural corpora unusable here. The cost is real: our passages are shorter and far more regular than a research paper's, the surrounding argument is absent, and a marker in a published proof carries authorial reputation that ours does not. We therefore claim the effect for controlled elision, not for published research mathematics. A naturally occurring probe set, with depth annotated by expert readers over real proofs, is the necessary complement and would test whether the deference we measure survives when the mathematics around it is genuine.

Depth and length move together. Naming d operations costs words. We control for length and repeat the contrast within length bands, but cannot fully dissociate the two.

The tier ordering. The assignment of markers to effort tiers is ours. We report it as a stipulated construct rather than a validated scale, and Appendix C gives the protocol for the annotation study that would validate it: readers see each marker in a fixed carrier sentence and rate, on a five-point scale, how much work the phrase claims was omitted, with tier recovered as the rank order of mean ratings. Two things limit the damage in the meantime. The contrast that carries the result is S_1 against S_3, whose ordering is not in serious doubt: *Clearly* claims less omitted work than *a somewhat lengthy computation shows*. And each tier draws on six paraphrases, so no single phrase carries the effect. What validation would buy is the ordinal spacing the regression assumes, which we cannot currently defend. Marker inventories also differ across languages and mathematical subcultures; all our items are English in one register.

Ethical considerations

ELIDE-BENCH is generated from symbolic templates and contains no personal data, no scraped copyrighted text, and no content about people. The finding has a dual-use edge: a system that accepts false mathematics more readily when the author calls the derivation long can be exploited by anyone wanting a weak step waved through. We report the mitigation alongside the effect.

References

Tanja Baeumel, Josef van Genabith, and Simon Ostermann. 2025. Disentangling mathematical reasoning in LLMs: A methodological investigation of internal mechanisms. In *Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP)*. Association for Computational Linguistics.

Nicolaas Govert de Bruijn. 1994. The mathematical vernacular, a language for mathematics with typed sets. In *Selected Papers on Automath*. North-Holland.

Carolina Dias-Alexiou, Edison Marrese-Taylor, and Yutaka Matsuo. 2025. An in-depth human study of the mathematical reasoning abilities in large language models. In *Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP)*. Association for Computational Linguistics.

Deborah Ferreira and André Freitas. 2020. Natural language premise selection: Finding supporting statements for mathematical text. In *Proceedings of the 12th Language Resources and Evaluation Conference (LREC)*.

Mohan Ganesalingam. 2013. *The Language of Mathematics: A Linguistic and Philosophical Investigation*. Lecture Notes in Computer Science. Springer.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, et al. 2020. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

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

Ken Hyland. 1998. *Hedging in Scientific Research Articles*. John Benjamins.

George Lakoff. 1973. Hedges: A study in meaning criteria and the logic of fuzzy concepts. *Journal of Philosophical Logic*, 2(4).

Zichao Li. 2025. Formula-text cross-retrieval: A benchmarking study of dense embedding methods for mathematical information retrieval. In *Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP)*. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*.

Yen-Ting Lin, Di Jin, Tengyu Xu, Tianhao Wu, Sainbayar Sukhbaatar, Chen Zhu, Yun He, Yun-Nung Chen, Jason E Weston, Yuandong Tian, Arash Rahnama, Sinong Wang, Hao Ma, and Han Fang. 2025. Step-KTO: Optimizing mathematical reasoning through stepwise binary feedback. In *Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP)*. Association for Computational Linguistics.

Franklin Ma and Alan J. Hu. 2025. Logically constrained decoding. In *Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP)*. Association for Computational Linguistics.

Yuta Matsumoto, Benjamin Heinzerling, Masashi Yoshikawa, and Kentaro Inui. 2022. Tracing and manipulating intermediate values in neural math problem solvers. In *Proceedings of the 1st Workshop on Mathematical Natural Language Processing (MathNLP)*. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, Amit Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, et al. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3.

Microsoft. 2025. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2025. GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations (ICLR)*.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Anja Reusch and Wolfgang Lehner. 2022. Extracting operator trees from model embeddings. In *Proceedings of the 1st Workshop on Mathematical Natural Language Processing (MathNLP)*. Association for Computational Linguistics.

Elisaveta Samoylov and Soroush Vosoughi. 2025. Modeling tactics as operators: Effect-grounded representations for Lean theorem proving. In *Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP)*. Association for Computational Linguistics.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R. Bowman, Esin Durmus, Zac Hatfield-Dodds, Scott R. Johnston,

Shauna Kravec, Timothy Maxwell, Sam McCandlish, Kamal Ndousse, Oliver Rausch, Nicholas Schiefer, Da Yan, Miranda Zhang, and Ethan Perez. 2024. Towards understanding sycophancy in language models. *International Conference on Learning Representations (ICLR)*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning (ICML)*.

Gladys Tyen, Hassan Mansoor, Victor Cărbune, Peter Chen, and Tony Mak. 2024. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Ulme Wennberg and Gustav Eje Henter. 2024. Exploring internal numeracy in language models: A case study on ALBERT. In *Proceedings of the 2nd Workshop on Mathematical Natural Language Processing*. ELRA and ICCL.

Dane A. Williamson, Yangfeng Ji, and Matthew B. Dwyer. 2025. Syntactic blind spots: How misalignment leads to LLMs' mathematical errors. In *Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP)*. Association for Computational Linguistics.

A Additional Results

This appendix holds the two tables referenced from Section 6 and Section 7 that did not fit in the main text, together with a visual summary of the intervention results.

Model	Q1	Q2	Q3	Q4
Qwen3-4B-Instr	−0.80	−0.26	+0.14	−0.89
Phi-4-mini-Instr	−1.00	−0.83	−0.94	−1.00

Table 6: Depth effect inside quartiles of passage length, as the Spearman correlation between elision depth and false acceptance within each band. Naming d operations costs words, so depth and length are correlated by construction; the correlations here are negative or near zero throughout, which matches the negative β_d rather than overturning it.

Model	Method	Bal.	Det.	β_d	$\Delta_{S_3-S_1}$
QWEN3-4B	direct	0.523	0.988	+0.024	−0.050
	marker masked	0.517	1.000	−0.083	+0.000
	CoT (matched)	**0.685**	0.975	−0.648	−0.033
	EAV B=2	0.544	0.996	−0.179	+0.000
	EAV B=4	0.567	0.962	−0.152	−0.067
	EAV B=8	0.608	0.921	−0.017	−0.067
	EAV B=8 sym	0.608	0.921	−0.017	−0.067
	EAV B=8 mask	0.608	0.933	−0.298	+0.000
PHI-4-MINI	direct	0.535	0.992	+0.032	−0.033
	marker masked	0.517	1.000	+0.015	+0.000
	CoT (matched)	0.631	0.900	−0.396	+0.000
	EAV B=2	0.540	0.742	−0.470	−0.050
	EAV B=4	0.552	0.633	−0.382	+0.017
	EAV B=8	0.604	0.788	−0.582	−0.017
	EAV B=8 sym	0.602	0.788	−0.582	−0.017
	EAV B=8 mask	**0.633**	0.800	−0.790	+0.000

Table 7: EAV and ablations on the 60-family generative subset, full budget sweep. The token-matched chain-of-thought baseline matches or beats every EAV variant on both models, so the gain is attributable to inference-time computation rather than to expanding the elided block. Marker masking drives the rhetoric gap $\Delta_{S_3-S_1}$ to zero in both models.

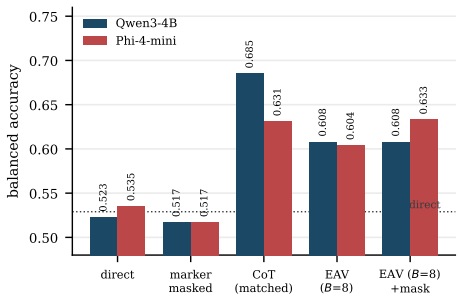


Figure 4: EAV against a chain-of-thought baseline matched for generation budget. The matched baseline is at least as good as EAV on both models (Section 7).

B Marker Inventory

Each family samples one marker uniformly from its tier, so no result is driven by a single phrase. S_0 juxtaposes the claim with no marker and is the reference condition in the regression, which uses the ordinal contrast over S_1 to S_3 only.

Tier	Markers
S_0	(no marker; bare juxtaposition)
S_1	*Clearly,* *Obviously,* *Trivially,* *It is immediate that* *It is easy to see that* *One sees at once that*
S_2	*It follows that* *Hence,* *Thus,* *Consequently,* *This gives* *We therefore obtain that*
S_3	*A somewhat lengthy computation shows that* *After a careful case analysis, one finds that* *With some work, one can check that* *A routine but tedious calculation yields* *Expanding and regrouping repeatedly, one obtains* *After several pages of bookkeeping, one arrives at*

Table 8: Marker inventory by claimed-effort tier.

C Tier Validation Protocol

Each of the 18 markers appears in a fixed carrier sentence, so only the marker varies: *"Let* $n \equiv 5 \pmod{17}$. *[marker]* $n^2 \equiv 8 \pmod{17}$." Annotators rate *how much work the author claims to have left out*, from 1 (none) to 5 (a great deal), and are told the question concerns omitted work, not truth. Markers appear in random order with three repeats for within-annotator reliability. Five annotators suffice, since the quantity of interest is the rank order of mean ratings, predicted to be $S_1 < S_2 < S_3$; agreement is Krippendorff's α.

AMIR-GRPO: Inducing Implicit Preference Signals into GRPO

Amir Hossein Yari **Fajri Koto**
Department of Natural Language Processing, MBZUAI
{amir.yari,fajri.koto}@mbzuai.ac.ae
https://github.com/AmirHosseinYari2002/AMIR-GRPO

Abstract

Despite its effectiveness for post-training reasoning models, group relative policy optimization (GRPO) suffers from several key limitations. Specifically, it introduces response-level length bias, weakly penalizes low-quality trajectories, and reduces rich intra-group preference information to a simplified scalar reward, limiting direct pairwise comparison between reasoning trajectories. To address these issues, we propose AMIR-GRPO, which augments GRPO with an implicit DPO-style contrastive regularizer constructed directly from intra-group reward rankings, without requiring additional annotations. This mechanism strengthens suppression of low-reward trajectories, mitigates length bias, and converts each rollout group into a denser set of supervision constraints. Across multiple mathematical reasoning benchmarks, AMIR-GRPO consistently outperforms strong GRPO baselines, achieves clearer separation between correct and incorrect reasoning chains, and delivers broader coverage beyond instances solved by standard GRPO.

1 Introduction

Post-training has become the primary mechanism for scaling the reasoning capabilities of large language models (LLMs). Unlike supervised fine-tuning, reinforcement learning (RL) from outcome-level feedback directly optimizes for high-quality reasoning behavior (OpenAI, 2024; Guo et al., 2025). Within this paradigm, group relative policy optimization (GRPO) (Shao et al., 2024) has emerged as a popular approach: it samples multiple completions per query and normalizes rewards within each group, producing a low-variance policy-gradient estimator while eliminating the need for a separate value network (Shao et al., 2024; Zheng et al., 2025).

In parallel, preference-based objectives such as Direct Preference Optimization (DPO) (Rafailov et al., 2023) effectively sharpen decision boundaries by contrasting preferred and rejected responses through pairwise supervision. However, these approaches fundamentally rely on large curated preference datasets, which are expensive to construct—especially for mathematical reasoning, where high-quality labels are scarce.

These paradigms offer complementary strengths: outcome-based reinforcement learning scales through absolute rewards, while contrastive learning efficiently leverages relative quality signals. However, standard GRPO does not fully exploit the ordering it induces within each rollout group. A rollout group of size G induces up to $O(G^2)$ pairwise preference relations, yet standard GRPO collapses this rich signal into only $O(G)$ scalar advantages. We argue that incorporating explicit contrastive objectives into GRPO could unlock these latent signals, improving sample efficiency and strengthening reasoning alignment.

Furthermore, specific design choices in GRPO limit its efficacy in shaping reasoning. First, assigning a single trajectory-level advantage uniformly to all tokens introduces a systematic length bias: with positive advantages, shorter answers receive larger per-token updates, implicitly favoring brevity even when extended reasoning is required; with negative advantages, penalties are diluted over long responses, weakening suppression of long, low-reward trajectories (Liu et al., 2025b). Second, in the sparse-reward regime characteristic of difficult reasoning tasks, group normalization yields weak penalties on incorrect trajectories. When few completions are correct and many are wrong, the negative advantage assigned to errors is diluted, failing to effectively suppress long, incorrect chains of thought. We argue that adding contrastive objectives to GRPO mitigates these issues by shifting optimization from uniform token-level updates to relative completion-level comparisons. Contrastive signals strengthen penalties for low-quality reason-

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 69–85
October 29, 2026 ©2026 Association for Computational Linguistics

ing through pairwise comparisons and fully exploit the ordering within rollout groups—boosting sample efficiency and alignment with reasoning quality.

These observations lead to our primary research question: *Can we enrich group-based policy optimization with contrastive, preference-like structure—without requiring explicit pairwise labels while preserving GRPO's scalability?* We answer this question affirmatively by augmenting GRPO with an implicit preference objective derived directly from its own rollouts.

We introduce ***Augmented Merit-guided Implicit-preference Regularization for GRPO*** (AMIR-GRPO), which injects DPO-style contrastive supervision into GRPO while preserving its simplicity and off-policy compatibility. AMIR-GRPO combines GRPO's scalability in environment-driven rollouts with the boundary-sharpening effect of contrastive preference learning. The core idea is to reinterpret reward orderings within each rollout group as implicit pairwise preferences: for a fixed query, lower-reward completions act as rejected outputs and are explicitly contrasted against higher-reward completions. This turns each low-reward rollout into a first-class negative example, enabling the policy not only to reinforce effective reasoning patterns but also to more directly suppress undesirable ones. Consequently, AMIR-GRPO mitigates response-level length bias by enforcing negative gradients on long, low-reward trajectories, and it amplifies penalties in sparse-reward settings by converting each low-reward completion into multiple contrastive negatives. Empirically, this framework yields less collapsed, more coverage-enhancing reasoning behavior than standard GRPO, together with a sharper decision boundary.

Across multiple mathematical benchmarks, AMIR-GRPO consistently improves reasoning-focused post-training over strong GRPO baselines. Models trained with AMIR-GRPO achieve higher accuracy, exhibit clearer separation between correct and incorrect trajectories, and show reduced response-length bias. Our analysis suggests that standard GRPO mainly reweights the base model's existing reasoning modes, whereas AMIR-GRPO better preserves diversity while increasing alignment with high-quality chains. Notably, AMIR-GRPO expands effective competence: rather than only redistributing probability over already-solved instances, it solves additional problems that remain unsolved by both the base model and a tuned GRPO model.

2 Methodology

2.1 Preliminaries

GRPO is a PPO-style algorithm that enables value-free policy learning through group-relative credit assignment. For each query q, the behavior policy $\pi_{\theta_{\text{old}}}$ samples a group of G candidate responses $\{o_i\}_{i=1}^{G}$ with scalar rewards $\{r_i\}_{i=1}^{G}$. The normalized intra-group advantage for response i is defined as

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}\Big(\{r_j\}_{j=1}^{G}\Big)}{\text{std}\Big(\{r_j\}_{j=1}^{G}\Big)}.$$

which is then shared across all tokens along its trajectory. Using these advantages, GRPO maximizes the clipped surrogate objective in Eq. (1). The expectation in $\mathcal{J}_{\text{GRPO}}(\theta)$ is taken over queries q and response groups $\{o_i\}_{i=1}^{G}$ generated by the behavior policy $\pi_{\theta_{\text{old}}}$. For each response o_i, the objective averages over token positions t and applies PPO-style clipping to the importance ratio

$$\frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}$$

within the interval $[1-\varepsilon, 1+\varepsilon]$, scaled by $\hat{A}_{i,t}$. This retains PPO's trust-region-like behavior while replacing a parametric critic with group-normalized rewards.

GRPO further adds an explicit KL regularization term between the current policy π_θ and a reference policy π_{ref}, weighted by γ in Eq. (1). The corresponding divergence is estimated at the token level using the non-negative, unbiased Monte Carlo estimator of Schulman (2020):

$$\widehat{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}) = r_{i,t} - \log r_{i,t} - 1$$

where

$$r_{i,t} = \frac{\pi_{\text{ref}}\big(o_{i,t} \mid q, o_{i,<t}\big)}{\pi_\theta\big(o_{i,t} \mid q, o_{i,<t}\big)}$$

DPO trains a target policy π_θ to match human preference judgments while remaining close to a fixed reference policy π_{ref}. Training proceeds in two stages. For each query q, we first sample a pair of candidate completions $y_1, y_2 \sim \pi_{\text{ref}}(\cdot \mid q)$ from the reference policy and obtain a human label indicating which completion is preferred. This yields an offline preference dataset

$$\mathcal{D} = \Big\{\big(q^{(i)}, y_{\text{p}}^{(i)}, y_{\text{r}}^{(i)}\big)\Big\}_{i=1}^{N}$$

where $y_{\text{p}}^{(i)}$ and $y_{\text{r}}^{(i)}$ denote the preferred and rejected responses to the i-th query, respectively. Given $\mathcal{D}$, the frozen reference policy π_{ref}, and a scale parameter β, DPO optimizes π_θ by maximizing the

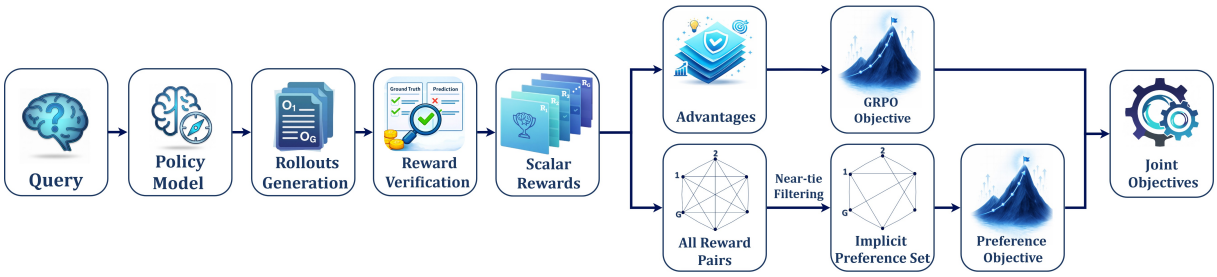


Figure 1: End-to-end training pipeline of AMIR-GRPO.

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\big[q \sim P(Q),\ \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(O \mid q)\big] \\ \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}\hat{A}_{i,t},\ \text{clip}\left(\frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right] - \gamma\,\mathbb{D}_{\text{KL}}\big[\pi_\theta \,\|\, \pi_{\text{ref}}\big]\right\} \tag{1}$$

objective $\mathcal{J}_{\text{DPO}}$ in Eq. (2). For each triple (q, y_p, y_r), the argument of the logistic function $\sigma(\cdot)$ is the β-scaled difference between the log-likelihood ratios of the preferred and rejected completions, both measured relative to π_{ref}. Maximizing $\mathcal{J}_{\text{DPO}}$ therefore upweights preferred responses and downweights rejected ones, while β acts as a temperature-like hyperparameter that controls how far π_θ is allowed to move away from π_{ref} to fit the observed preferences, balancing preference alignment against proximity to the reference model.

2.2 AMIR-GRPO

AMIR-GRPO augments GRPO by leveraging relative supervision within each rollout group. Rather than treating rewards as isolated scalars, we interpret their within-group ranking as implicit pairwise feedback and use it to refine the policy update. Figure 1 provides a schematic of AMIR-GRPO: group-wise rewards induce an implicit preference set, which is enforced by a DPO-style contrastive term and optimized jointly with GRPO.

Given a query $q \sim P(Q)$, GRPO samples a group of G completions $\{o_i\}_{i=1}^{G}$ with scalar rewards $\{r_i\}_{i=1}^{G}$. A naive construction would include all ordered pairs, yielding $O(G^2)$ preferences per group. To discard near ties and retain only reliable comparisons, we define the implicit preference set

$$\mathcal{S}(q) = \big\{(i,j) \,\big|\, r_i > r_j,\ r_i - r_j > \delta_r\big\}$$

where $\delta_r > 0$ is a reward margin. Each completion may participate in multiple comparisons, so the information within each group of rollouts is exploited more thoroughly. In particular, low-reward completions appear repeatedly as the negative side of the comparison, amplifying their influence beyond what a single scalar advantage can express and leading to stronger suppression of consistently poor reasoning paths.

For each $(i,j) \in \mathcal{S}(q)$ we treat o_i as preferred and o_j as rejected, and introduce the DPO-style logit

$$z_{i,j}(\theta) = \beta_{\text{DPO}}\Big[\big(\ell_\theta(q,o_i) - \ell_{\text{ref}}(q,o_i)\big) - \big(\ell_\theta(q,o_j) - \ell_{\text{ref}}(q,o_j)\big)\Big]$$

where $\ell_\theta(q,o)$ and $\ell_{\text{ref}}(q,o)$ denote the length-normalized log-probabilities of completion o under the current and reference policies, and β_{DPO} is a temperature. The associated preference regularizer is

$$\mathcal{J}_{\text{pref}}(\theta) = \mathbb{E}_{(q,i,j)\sim\mathcal{S}}\big[\log\sigma\big(z_{i,j}(\theta)\big)\big]$$

where $\mathcal{S} = (q,i,j) \mid (i,j) \in \mathcal{S}(q)$ is the union of all per-query preference sets. In practice, for a mini-batch of queries we approximate this expectation by a simple average over all mined pairs in the batch

$$\mathcal{J}_{\text{pref}}(\theta) \approx \frac{1}{\sum_q |\mathcal{S}(q)|}\sum_{q}\sum_{(i,j)\in\mathcal{S}(q)}\log\sigma\big(z_{i,j}(\theta)\big)$$

so that maximizing $\mathcal{J}_{\text{pref}}$ encourages the policy to assign higher log-probability to preferred completions than to rejected ones, with an effective margin controlled by β_{DPO}.

The overall AMIR-GRPO objective augments the standard GRPO loss with this preference term:

$$\mathcal{J}_{\text{AMIR-GRPO}}(\theta) = \mathcal{J}_{\text{GRPO}}(\theta) + \lambda_{\text{reg}}\,\mathcal{J}_{\text{pref}}(\theta)$$

where $\lambda_{\text{reg}} > 0$ scales the contrastive regularizer. To avoid underusing the preference signal or letting it dominate the GRPO objective, we adapt λ_{reg} online to maintain a stable ratio between their contributions. Specifically, after computing the GRPO loss and the contrastive preference loss on each mini-batch, we track the regularizer's relative con-

$$\mathcal{J}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = \mathbb{E}_{(q,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_p|q)}{\pi_{\text{ref}}(y_p|q)} - \beta \log \frac{\pi_\theta(y_r|q)}{\pi_{\text{ref}}(y_r|q)}\right)\right] \quad (2)$$

tribution to the GRPO objective and adjust λ_{reg} multiplicatively to keep this ratio within $[0.30, 0.70]$. When the preference term is underrepresented, λ_{reg} is increased; when it becomes too dominant, λ_{reg} is reduced. This prevents the model from drifting toward pure preference fitting while preserving a consistent preference-guided learning signal.

3 Experimental Setup

3.1 Datasets

Our models are trained on the training split of GSM8K (Cobbe et al., 2021), a benchmark of grade-school word problems designed to probe multi-step mathematical reasoning.

For evaluation, we employ a broad suite of math-centric benchmarks: the GSM8K test split (Cobbe et al., 2021), the 2025 American Invitational Mathematics Examination (AIME 2025) (Zhang and Math-AI, 2025), OlympiadBench (He et al., 2024), AMC23 (math-ai), Minerva (Lewkowycz et al., 2022), AQUA-RAT (Ling et al., 2017), and Live-MathBench (Liu et al., 2025a). Collectively, these datasets span diverse difficulty levels and problem formats, allowing us to assess generalization beyond GSM8K and robustness under more challenging mathematical settings.

3.2 Training Protocol

We evaluate our approach by fine-tuning Qwen2.5-Instruct (3B and 7B) (Qwen et al., 2025) and Gemma3-it (4B) (Team et al., 2025) under a unified training configuration, comparing the standard GRPO objective with our AMIR-GRPO variant. We apply LoRA with rank 16 and scaling factor $\alpha = 16$, a maximum sequence length of 2048 tokens, a learning rate of 5×10^{-6}, and the AdamW optimizer, and use a completion group size of 8.

The reward functions provide the learning signal for our policy optimization. We employ three complementary components: (i) a *correctness reward* that verifies whether the final prediction matches the ground-truth answer, (ii) a *format reward*, which encourages adherence to the prescribed structured output format; and (iii) a *calibration reward*, which incentivizes well-calibrated confidence estimates. A trustworthy prediction system should produce confidence scores that are calibrated, i.e., the reported confidence should coincide with the true probability that the prediction is correct. To operationalize this, we use a *Verbalized Confidence* protocol (Tian et al., 2023): for each question–response pair, the model is prompted to output both an answer and an explicit confidence value $q \in [0, 1]$. We then calibrate this score by minimizing a strictly proper scoring rule L with respect to the binary correctness indicator $a \in \{0, 1\}$; in expectation, S is uniquely minimized when the reported confidence matches the true success probability $p(a = 1)$:

$$\mathbb{E}_{a\sim p(a)}\big[L(p(a), a)\big] \;\le\; \mathbb{E}_{a\sim p(a)}\big[L(q, a)\big]$$

In our implementation, we instantiate S as the Brier score (Glenn et al., 1950):

$$L(q, a) = (a - q)^2$$

where $a \in \{0, 1\}$ indicates correctness. The resulting *calibration reward* is defined as its complement, $1 - L(q, a)$. The overall reward R is then given by a weighted sum of the three components:

$$R \;=\; w_{\text{corr}}\, r_{\text{corr}} \;+\; w_{\text{fmt}}\, r_{\text{fmt}} \;+\; w_{\text{calib}}\, r_{\text{calib}}$$ [1]

We selected the reward weights and preference-construction threshold from the semantics of the reward decomposition. Appendix B.1 provides the full rationale for the reward weights and margin.

3.3 Evaluation Metric

Conventional evaluation protocols aggregate scores from greedy decoding or nucleus sampling (Holtzman et al., 2020), thereby reflecting only average-case behavior. This can underestimate a model's true capability: a system may fail on a difficult instance under a small sampling budget even though additional samples would yield a correct solution. To address this, we adopt the Pass@k metric.

A naive computation of Pass@k from exactly k samples per problem suffers from high variance. We therefore employ the unbiased estimator introduced by Chen et al. (2021), which measures, for a given question q, the probability that at least one correct solution appears among k sampled generations:

$$\text{Pass@}k = \mathbb{E}_q\left[1 - \frac{\binom{n-c_q}{k}}{\binom{n}{k}}\right]$$

where n is the total number of generations produced for question q, and c_q is the number of correct generations among them. Consequently, Pass@k

[1] In our experiments $(w_{\text{corr}}, w_{\text{fmt}}, w_{\text{calib}}) = (2.0, 0.9, 1.0)$

Dataset	k	Qwen2.5-3B-Instruct	Qwen2.5-7B-Instruct	Gemma-3-4B-it
GSM8K	1	83.6 → **83.4** (-0.2)	90.4 → **90.2** (-0.2)	86.2 → **86.4** (+0.2)
	2	90.2 → **89.9** (-0.3)	94.6 → **94.3** (-0.3)	90.3 → **90.7** (+0.4)
	4	93.6 → **93.4** (-0.2)	96.4 → **96.2** (-0.2)	92.8 → **93.2** (+0.4)
AIME25	1	0.8 → **2.1** (+1.3)	4.2 → **5.8** (+1.6)	3.7 → **5.8** (+2.1)
	2	1.7 → **4.2** (+2.5)	7.5 → **9.6** (+2.1)	4.9 → **8.7** (+3.8)
	4	3.3 → **8.3** (+5.0)	12.1 → **13.8** (+1.7)	5.9 → **12.4** (+6.5)
OlympiadBench	1	26.4 → **27.0** (+0.6)	36.2 → **36.4** (+0.2)	22.2 → **22.7** (+0.5)
	2	33.8 → **34.5** (+0.7)	43.6 → **43.7** (+0.1)	27.3 → **28.1** (+0.8)
	4	40.9 → **42.4** (+1.5)	50.5 → **50.6** (+0.1)	31.6 → **33.0** (+1.4)
AMC23	1	30.9 → **32.8** (+1.9)	45.0 → **47.8** (+2.8)	29.4 → **35.0** (+5.6)
	2	40.5 → **43.2** (+2.7)	55.4 → **60.9** (+5.5)	35.4 → **41.8** (+6.4)
	4	52.3 → **53.5** (+1.2)	63.7 → **73.1** (+9.4)	39.2 → **48.2** (+9.0)
MinervaMath	1	25.2 → **26.3** (+1.1)	35.2 → **35.9** (+0.7)	20.9 → **21.9** (+1.0)
	2	32.6 → **33.7** (+1.1)	41.6 → **42.8** (+1.2)	26.9 → **28.3** (+1.4)
	4	38.9 → **40.3** (+1.4)	46.5 → **48.4** (+1.9)	32.3 → **33.7** (+1.4)
AQUA-RAT	1	52.6 → **54.0** (+1.4)	59.0 → **60.0** (+1.0)	50.8 → **51.9** (+1.1)
	2	61.0 → **62.0** (+1.0)	66.0 → **66.5** (+0.5)	56.4 → **57.5** (+1.1)
	4	66.7 → **67.3** (+0.6)	70.4 → **70.8** (+0.4)	61.4 → **62.3** (+0.9)
LiveMathBench	1	19.0 → **23.1** (+4.1)	27.4 → **30.2** (+2.8)	13.9 → **15.8** (+1.9)
	2	24.1 → **31.6** (+7.5)	34.5 → **38.4** (+3.9)	17.4 → **19.0** (+1.6)
	4	27.3 → **39.2** (+11.9)	41.7 → **45.1** (+3.4)	20.9 → **22.1** (+1.2)

Table 1: Comparative performance on math benchmarks. Numbers are reported in %; Cells are formatted as GRPO → **AMIR-GRPO** (+gain).

corresponds to the expected fraction of questions for which at least one of the k sampled generations is correct.

Unless stated otherwise, we fix $n = 8$ generations per question and report Pass@k for $k \in \{1, 2, 4\}$.[2]

4 Results and Analysis

Table 1 summarizes the main results. Across all evaluated values of k, AMIR-GRPO consistently improves out-of-distribution generalization while maintaining GRPO-comparable performance on the in-distribution GSM8K benchmark, yielding a robust balance between accuracy at small k and exploration at large k.

These findings support our core claim: implicit-preference regularization provides richer supervision at no additional data cost, increasing the value extracted from existing datasets. Appendix B details the training dynamics, while Appendix A provides the full prompt template.

4.1 β_{DPO} Sensitivity

In the AMIR-GRPO objective, β_{DPO} acts as a temperature for the DPO-style contrastive preference regularizer $\mathcal{J}_{\text{pref}}(\theta)$, controlling the effective margin by which the policy distinguishes high- from low-reward completions. Adding $\mathcal{J}_{\text{pref}}(\theta)$ consistently improves performance, but the magnitude of the gain is sensitive to β_{DPO} and is jointly governed by the base model's capacity, initial competence and the dataset's difficulty.

For small, initially weak models (e.g., Qwen-3B), smaller β_{DPO} is preferable: limited capacity makes them prone to high-variance preference gradients, so a gentle contrastive signal stabilizes training and prevents $\mathcal{J}_{\text{pref}}$ from dominating the core $\mathcal{J}_{\text{GRPO}}$ term. Larger models with higher capacity but similarly low initial competence (e.g., Qwen-7B) tolerate stronger alignment pressure and typically peak at moderate β_{DPO} , where the induced margin is decisive without becoming overconfident. For already strong base models (e.g., Gemma-4B), higher β_{DPO} tends to be more beneficial.

Even within a fixed architecture, the optimal value remains dataset-dependent. For our 7B policy, a moderate β_{DPO} performs best on average, yet the most demanding benchmarks (e.g., AIME25 and LiveMathBench) favor larger settings—suggesting that a wider preference margin helps capitalize on subtle reward differences in complex regimes. Overall, these results underscore the need to tune β_{DPO} jointly to the model and data to fully unlock AMIR-GRPO's alignment benefits; the corresponding trends are summarized in Figure 2.

4.2 Are RL Reasoning Paths Truly Novel?

A recurring question in RL-based reasoning is whether performance gains stem from truly novel reasoning trajectories or from reweighting trajectories that were already plausible under the base model. Yue et al. (2025a) study this by analyzing conditional perplexity of generated responses and conclude that reinforcement learning largely sharpens probability mass around pre-existing modes rather than discovering fundamentally new reasoning patterns. In the same vein, we examine how AMIR-GRPO reshapes the distribution of reasoning chains relative to GRPO, using perplexity as a diagnostic of how well a sampled rollout is supported by the very policy that generated it.

Formally, for a model m, a problem x, and a response $Y = (y_1, \ldots, y_T)$ of length T tokens, we define conditional perplexity as the exponentiated average negative log-likelihood of Y under m (Yue et al., 2025a):

$$\text{PPL}_m(Y \mid x) = \exp\left(-\frac{1}{T}\sum_{t=1}^{T} \log P_m(y_t \mid x, y_1, \ldots, y_{t-1})\right)$$

Lower values of $\text{PPL}_m(Y \mid x)$ signify that Y lies deeper in high-density regions of m's conditional distribution, i.e., a more self-consistent and strongly

[2] Generations are sampled with temperature 0.6 and top-p sampling with $p = 1.0$.

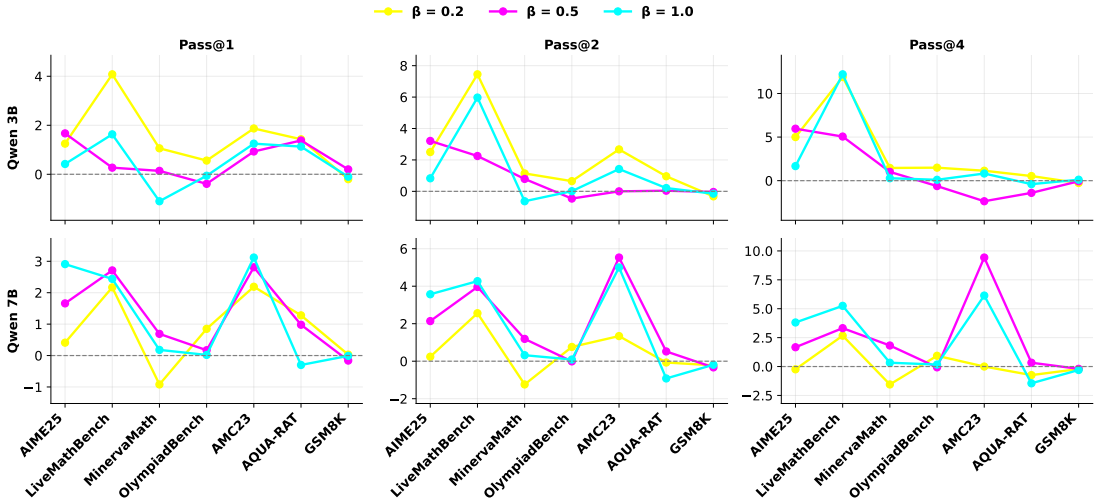


Figure 2: Impact of β_{DPO} across datasets. Values indicate $\Delta\text{Pass@}k$ relative to the GRPO baseline.

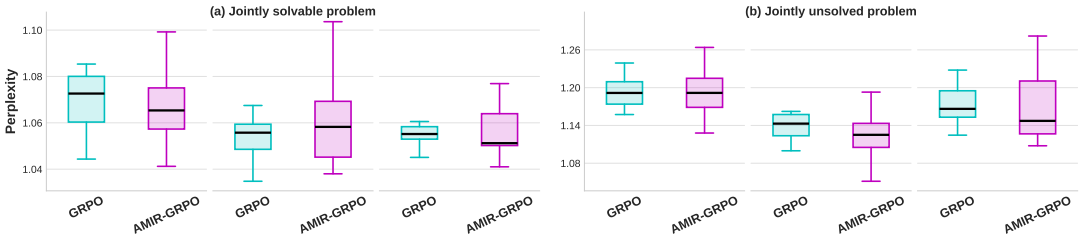


Figure 3: Comparison of reasoning-trajectory perplexity on an AMC23 question.

supported trajectory.

Using Gemma-4B, we analyze six AMC23 problems randomly selected from two pre-filtered strata: jointly solvable instances, where both tuned models (GRPO and AMIR-GRPO) consistently produce correct chains of thought across samples, and jointly unsolved instances, where all sampled trajectories are incorrect for both models. For each problem and each model, we sample 16 rollouts and compute $\text{PPL}_m(Y \mid x)$ under the same model m that generated Y. Figure 3 demonstrates the resulting distributions. As expected, jointly unsolved problems exhibit higher perplexity than jointly solvable ones, reflecting the greater challenge of generating chains that are internally supported in this regime.

Across both panels, GRPO typically produces tighter boxes with shorter whiskers, suggesting that it samples from a comparatively narrow band of rollouts that are similarly likely under its own policy—consistent with a more sharply mode-concentrated trajectory distribution. In contrast, AMIR-GRPO generally matches or slightly improves the median perplexity relative to GRPO, while exhibiting a wider interquartile range and longer whiskers. In other words, AMIR-GRPO often includes rollouts that are as self-likely as GRPO's typical samples (low perplexity), yet it also permits a nontrivial fraction of trajectories that the policy itself assigns lower probability (higher perplexity).

Because perplexity is evaluated under the generating model, variation in PPL_m directly reflects how peaked the policy is over the trajectories it actually produces. GRPO's tight clustering is consistent with strong RL-induced sharpening: once a small set of high-reward trajectories is identified, probability mass concentrates around a narrow mode, making most sampled chains uniformly high-likelihood under the GRPO policy. AMIR-GRPO retains some of this sharpening—its medians are frequently comparable or lower—but its regularization term curbs excessive collapse. The resulting distribution is effectively tempered: AMIR-GRPO can prioritize effective reasoning patterns (low median perplexity) while sustaining a thicker tail and broader interquartile range (greater trajectory diversity). This behavior is the expected signature of a regularized objective: it shifts mass toward stronger chains without extinguishing alternative, lower-probability reasoning paths that support exploration and coverage.

4.3 Solvable Problem Space

To further investigate whether RL training truly enlarges the set of solvable tasks or primarily reweights

trajectories already available to the Base model (Qwen-3B). On AMC23, we define a question as solvable if at least one of 16 sampled rollouts yields the correct answer, and compare solvable sets across the Base model and its GRPO- and AMIR-GRPO-trained variants. As summarized in Table 2, 70.6% of solvable questions lie in the intersection of all three systems, revealing a substantial shared core of problems that remain accessible regardless of the downstream objective. This overlap suggests that a large portion of the observed RL gains stem from reshaping probability mass over trajectories that were already realizable under the Base model, rather than from discovering entirely new modes of reasoning.

The remaining problems, however, highlight a clear contrast between the two RL objectives. Only 2.9% of solvable questions are handled by the Base model and the standard GRPO variant but not by AMIR-GRPO, and there are no questions uniquely solved by GRPO when both the Base and AMIR-GRPO fail. By contrast, AMIR-GRPO delivers a nontrivial expansion of coverage: 11.8% of solvable problems are addressed jointly by the Base and AMIR-GRPO but missed by GRPO, and an additional 8.8% are solved exclusively by AMIR-GRPO when both the Base and GRPO fail. Very few problems (2.9%) are solvable only by the Base model, indicating that neither RL configuration catastrophically erodes the Base model's capabilities.

Base	GRPO	AMIR-GRPO	Coverage
✓	✓	✓	70.6%
✓	✓	✗	2.9%
✓	✗	✓	11.8%
✓	✗	✗	2.9%
✗	✓	✓	2.9%
✗	✓	✗	0.0%
✗	✗	✓	8.8%
✗	✗	✗	0.0%

Table 2: Coverage of solvable AMC23 problems.

4.4 Sharper Margins, Stronger Reasoning

On Gemma-4B evaluated on LiveMathBench, we analyze a post-hoc preference margin defined as the per-problem difference between the mean length-normalized log-probability assigned to correct versus incorrect completions. This scalar margin is positive when the model, on average, assigns higher per-token likelihood to correct reasoning trajectories than to incorrect ones. Figure 4 shows that both GRPO and AMIR-GRPO achieve positive margins, but AMIR-GRPO yields a noticeably larger average margin (× 2.7) compared to GRPO, indicating a sharper and more confident separation between correct and incorrect solutions in probability space.

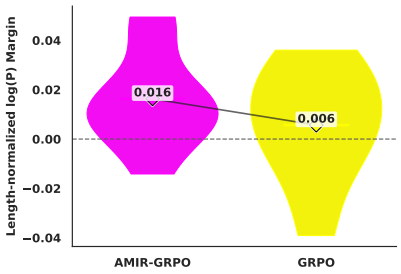


Figure 4: Preference margin separation.

4.5 Dissection of Reasoning Failures

We go beyond accuracy to characterize how models fail, conducting a fine-grained error analysis on AMC23. For each problem, we sample 16 rollouts from Qwen-7B and convert every solution into a stepwise form by segmenting it into numbered reasoning steps. For each incorrect rollout, we identify the first erroneous step[3] and assign an error label using our task-specific taxonomy (Appendix E). Labels are initially proposed by GPT-5 (OpenAI, 2025) and then manually audited to ensure consistency and avoid systematic misclassification.

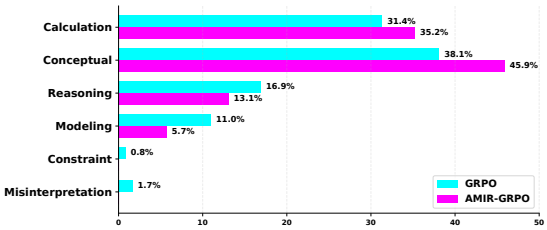


Figure 5: Error distribution.

Figure 5 summarizes the resulting error-type distribution, revealing the most prevalent failure modes. Complementarily, Figure 6 examines where breakdowns occur within the reasoning chain. For each incorrect solution, we compute a relative error position[4] and estimate a smoothed density over all failures. This tests whether errors tend to arise early (problem interpretation/formulation), mid-chain (intermediate inference), or late (final execution).

Comparing GRPO to AMIR-GRPO, Figure 6 shows a clear shift: AMIR-GRPO fails less frequently at both the start and the end of the chain. This suggests that AMIR-GRPO reduces

[3] the earliest point at which the reasoning irreversibly departs from a valid solution

[4] (step index)/(number of steps)

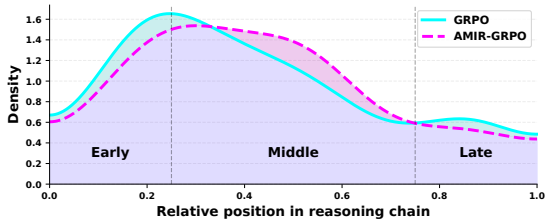


Figure 6: Failure locality density.

formulation-related errors (early-stage) and answer realization errors (late-stage), implying that its improvements reflect not only a higher success rate but also a systematic change in where reasoning tends to break when it fails.

4.6 Cross-Variant Integratability

Our approach is designed to integrate seamlessly with other GRPO-family variants, including GSPO (Zheng et al., 2025), offering robust compatibility. Notably, it delivers consistent gains across a wide suite of benchmarks, as reported in Table 3.

Dataset	Qwen2.5-7B-Instruct		
	Pass@1	Pass@2	Pass@4
GSM8K	90.3 → **90.5** (+0.2)	94.3 → **94.3** (+0.0)	95.8 → **96.0** (+0.2)
AIME25	3.7 → **7.1** (+3.4)	7.1 → **10.5** (+3.4)	12.9 → **14.3** (+1.4)
AMC23	45.6 → **47.5** (+1.9)	56.5 → **59.7** (+3.2)	66.2 → **70.1** (+3.9)
MinervaMath	35.0 → **35.5** (+0.5)	41.5 → **41.9** (+0.4)	46.2 → **47.5** (+1.3)
AQUA-RAT	60.2 → **61.0** (+0.8)	67.0 → **67.7** (+0.7)	71.3 → **71.8** (+0.5)
LiveMathBench	29.1 → **33.2** (+4.1)	37.6 → **41.3** (+3.7)	47.2 → **50.0** (+2.8)

Table 3: Comparative performance on benchmarks. Cells are formatted as: GSPO → **AMIR-GSPO** (+gain).

5 Related Work

RL has become a key driver of improved reasoning in LLMs. While PPO remains the standard choice, its actor–critic design requires a value network roughly the size of the policy, sharply increasing memory and compute (Schulman et al., 2017). This overhead has motivated critic-free, group-based alternatives that avoid explicit value estimation by exploiting relative rewards across multiple completions of the same prompt.

GRPO substitutes the explicit critic with a normalized reward baseline, delivering stable updates without an auxiliary value model (Shao et al., 2024). Subsequent variants refine how group-relative signals are computed and normalized. Dr.GRPO drops token-level averaging and reward normalization to mitigate length- and difficulty-induced optimization bias (Liu et al., 2025b). GPG similarly normalizes by the group mean to temper gradient variance with minimal overhead (Chu et al., 2025).

A complementary line of work focuses on where group-relative objectives act along the sequence and how they are adapted to long chain-of-thought (CoT) reasoning. GSPO lifts importance sampling and clipping from the token level to whole sequences, improving stability while retaining group-relative rewards (Zheng et al., 2025). SAPO replaces hard PPO-style clipping with a smooth, temperature-controlled trust region, yielding less brittle and more adaptive updates (Gao et al., 2025). TreeRPO extends GRPO to tree-structured reasoning by assigning group-relative rewards at intermediate nodes, converting sparse trajectory-level feedback into denser step-wise signals (Yang et al., 2025). DAPO introduces clip-Higher to preserve diversity and prevent entropy collapse, dynamic sampling to improve efficiency and stability, a token-level policy-gradient loss adapted to long sequences, and overlong reward shaping to suppress noise from excessively long outputs (Yu et al., 2025). Revisiting value-based designs, VAPO proposes a value-centric framework with length-adaptive generalized advantage estimation, improving advantage accuracy and stability while integrating several prior optimization refinements (Yue et al., 2025b).

Our method moves beyond GRPO variants that mainly adjust value modeling. We preserve GRPO's critic-free core and add an implicit preference objective defined directly from group rewards. This yields richer supervision at zero additional annotation cost, and integrates cleanly with GRPO-family extensions (e.g., GSPO).

6 Conclusion

This work identifies limitations of GRPO, showing that the standard formulation fails to fully exploit the within-group ordering present in rollout batches. We propose AMIR-GRPO, which augments GRPO with an implicit preference objective induced by groupwise rankings, while leaving the data collection pipeline unchanged. Across a spectrum of math-reasoning benchmarks, from school-level to Olympiad-level tasks, AMIR-GRPO consistently outperforms standard GRPO at a reasonable and affordable computational cost. Our analysis further indicates that the method sharpens decision boundaries and expands the set of solvable instances, rather than simply amplifying existing trajectories. Taken together, these results suggest that making implicit preference structure explicit in group-based RL provides a simple, scalable mechanism for strengthening reasoning supervision.

Limitations

Although AMIR-GRPO demonstrates strong empirical performance in our experiments, several limitations remain. First, we assess the method exclusively on text-based benchmarks. Accordingly, our conclusions are confined to textual reasoning; extending AMIR-GRPO to other modalities (e.g., code or vision–language reasoning) remains an important avenue for future work, where reward design, noise profiles, and failure modes may differ from those in purely textual tasks.

Second, whereas much prior work on long-horizon, verifiable reasoning focuses mainly on mathematical benchmarks (Nan et al., 2025; Zhu et al., 2025), we adopt a broader view by evaluating a wider range of math reasoning benchmarks than is typical. Even so, our evaluation still reflects a single stylistic regime. Broadening the study to additional reasoning domains—including commonsense (Sprague et al., 2024), cultural reasoning (Yari and Koto, 2025), and open-ended instruction following (Dussolle et al., 2025)—would better establish how general AMIR-GRPO's gains are under qualitatively different training signals and evaluation criteria.

Finally, our current formulation relies on trajectory-level outcome rewards without explicit process rewards or fine-grained credit assignment to intermediate reasoning steps. While this design choice simplifies the training pipeline, it may limit the algorithm's ability to correct early-step errors or to exploit partially correct reasoning traces. Compared with purely binary or final-answer rewards, incorporating richer intermediate signals to guide the reasoning trajectory could improve exploration efficiency and more precisely steer the model toward promising solution paths.

References

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Xiangxiang Chu, Hailang Huang, Xiao Zhang, Fei Wei, and Yong Wang. 2025. GPG: A simple and strong reinforcement learning baseline for model reasoning. *arXiv preprint arXiv:2504.02546*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Antoine Dussolle, A. Cardeña, Shota Sato, and Peter Devine. 2025. M-IFEval: Multilingual instruction-following evaluation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, Albuquerque, New Mexico. Association for Computational Linguistics.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. 2025. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*.

W Brier Glenn and 1 others. 1950. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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. 2024. 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)*, Bangkok, Thailand. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics.

Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang, Songyang Zhang, and Kai Chen. 2025a. Are your LLMs capable of stable reasoning? In *Findings of*

the Association for Computational Linguistics: ACL 2025, Vienna, Austria. Association for Computational Linguistics.

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

math-ai. Amc23. https://huggingface.co/datasets/math-ai/amc23.

Gongrui Nan, Siye Chen, Jing Huang, Mengyu Lu, Dexun Wang, Chunmei Xie, Weiqi Xiong, Xianzhou Zeng, Qixuan Zhou, Yadong Li, and 1 others. 2025. NGRPO: Negative-enhanced group relative policy optimization. *arXiv preprint arXiv:2509.18851*.

OpenAI. 2024. Learning to reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/.

OpenAI. 2025. Gpt-5 model documentation. https://platform.openai.com/docs/models/gpt-5.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36.

John Schulman. 2020. Approximating kl divergence. http://joschu.net/blog/kl-approx.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Zayne Rea Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2024. MuSR: Testing the limits of chain-of-thought with multistep soft reasoning. In *The Twelfth International Conference on Learning Representations*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. 2023. 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*, Singapore. Association for Computational Linguistics.

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Xiaodan Liang, Yiwei Wang, and Jing Tang. 2025. TreeRPO: Tree relative policy optimization. *arXiv preprint arXiv:2506.05183*.

Amir Hossein Yari and Fajri Koto. 2025. Unveiling cultural blind spots: Analyzing the limitations of mLLMs in procedural text comprehension. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vienna, Austria. Association for Computational Linguistics.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025a. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, and 1 others. 2025b. VAPO: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*.

Yifan Zhang and Team Math-AI. 2025. American invitational mathematics examination (aime) 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. 2025. The surprising effectiveness of negative reinforcement in LLM reasoning. *arXiv preprint arXiv:2506.01347*.

A Prompt Templates

We provide the prompt templates used in our experiments.

> A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The Assistant first thinks about the reasoning process in the mind, provides the user with the final answer, then analyzes its confidence about the solution and provides the user with its confidence level.
> The confidence level is a number between 0 and 1 (inclusive) enclosed within <confidence> </confidence> tags.
> The final answer is enclosed between <answer> </answer> tags.
> The analysis about confidence and uncertainty is enclosed within <analysis> </analysis> tags.
> The Assistant should reason about its confidence in the solution and its uncertainty in the solution within these tags.
> The final format that must be followed is:
> <think> reasoning process here </think>
> <answer> \boxed{final answer here} </answer>
> <analysis> analysis about confidence and uncertainty here </analysis>
> <confidence> confidence level here (number between 0 and 1) </confidence>

B Training Dynamics

Figure 7 summarizes the training dynamics of our models. Across all settings, AMIR-GRPO attains consistently lower Brier scores than vanilla GRPO, indicating better-calibrated confidence and a more trustworthy approach.

Its effect on response length is not uniform; it depends on the base model's capacity and initial competence, and acts as an adaptive regularizer that counterbalances GRPO's intrinsic bias. For low-capacity, low-competence models (Qwen-3B), it behaves like a pruning operator, sharply shortening outputs by suppressing long, low-reward hallucinations that GRPO's diluted scalar penalty fails to eliminate. Conversely, for competent models, AMIR-GRPO acts as a policy protector, preventing the detrimental brevity bias from compromising necessary reasoning depth. Specifically, it unshackles latent capacity in high-capacity, initially weaker models (Qwen-7B), allowing them to articulate the full Chain-of-Thought required for task mastery, thereby increasing length. Simultaneously, it stabilizes high-competence policies (Gemma-4B), preserving the robust structure of existing high-quality reasoning against GRPO's pressure to enforce premature token efficiency, which otherwise risks policy collapse. Thus, AMIR-GRPO provides an essential adaptive control: it curtails length only where it signifies policy error, yet preserves or expands it where it is indispensable for solution fidelity.

B.1 Hyperparameter Rationales

We choose the key hyperparameters from the semantics of the reward decomposition, rather than tuning them post hoc to favor AMIR-GRPO. As described in Section 3.2, the reward combines correctness, formatting, and calibration terms with weights

$$(w_{\text{corr}}, w_{\text{fmt}}, w_{\text{calib}}) = (2.0,\ 0.9,\ 1.0)$$

This weighting reflects the objective hierarchy of mathematical reasoning: correctness is the primary goal, while formatting and calibration serve as secondary regularizers. Since correctness contributes more than half of the maximum reward, well-formatted or well-calibrated but incorrect responses cannot be treated as desirable. Moreover, GRPO and AMIR-GRPO use the same reward structure, so the observed gains cannot be attributed to reward reweighting.

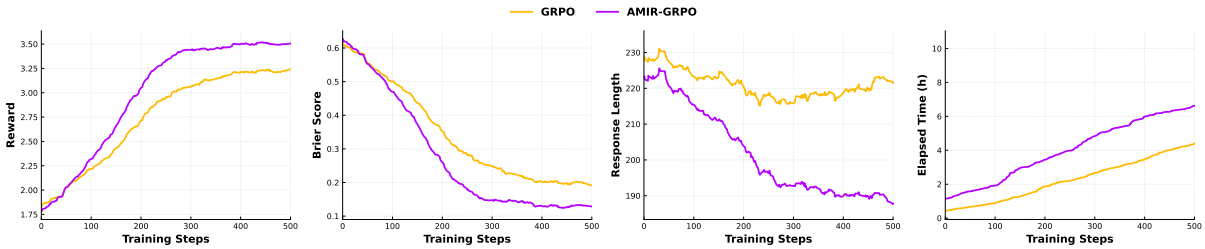
The preference margin is set to 2.0 directly from the reward design. The maximum reward is

$$2.0 + 0.9 + 1.0 = 3.9$$

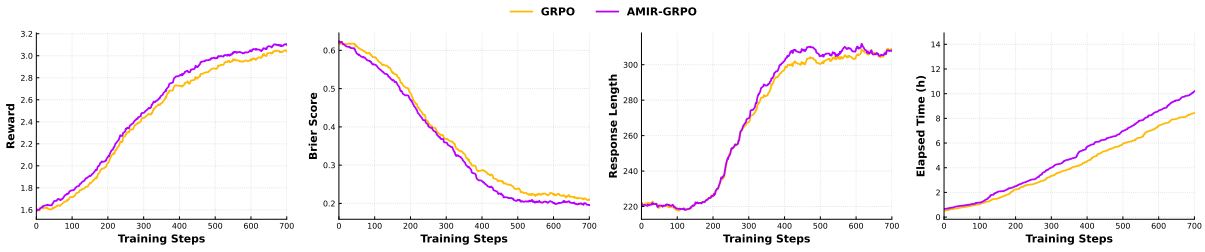
while formatting and calibration together can contribute at most 1.9. Thus, any response with reward at least 2.0 must include the correctness reward, whereas an incorrect response cannot exceed 1.9. The margin therefore provides a natural and unambiguous separation between desired and undesired samples, reducing label noise.

C Response Length Characteristics

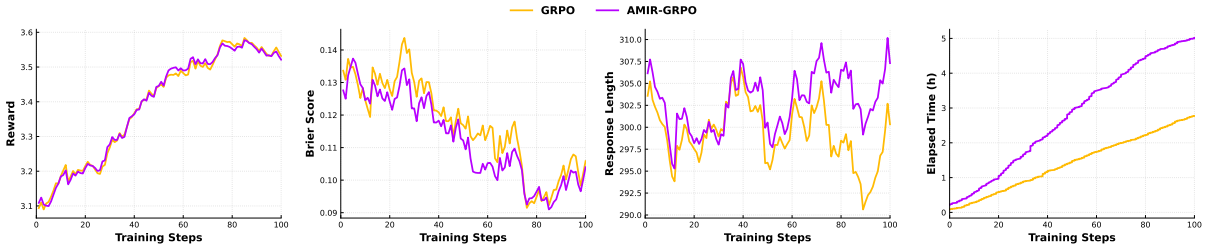
To quantify response-level length bias, we sample 16 rollouts per query from Qwen-3B on AIME25 and Qwen-7B on AMC23 after fine-tuning with GRPO or AMIR-GRPO. For AIME25, incorrect Qwen-3B trajectories average 747.7 tokens with GRPO versus 738.9 tokens with AMIR-GRPO. On AMC23, the corresponding averages for Qwen-7B are 781.4 and 759.4 tokens. These shorter low-



(a) Qwen-3B



(b) Qwen-7B



(c) Gemma-4B

Figure 7: Training behavior of GRPO vs. AMIR-GRPO.

reward trajectories under AMIR-GRPO support our claim that it mitigates response-level length bias by amplifying negative gradients on long, low-reward trajectories, thereby increasing their effective penalty.

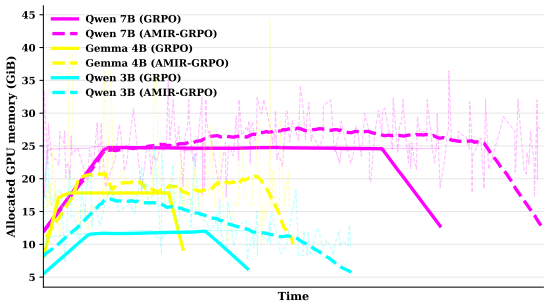


Figure 8: GPU memory footprint.

D Memory Footprint Analysis

Figure 8 indicates that AMIR-GRPO incurs only a slight GPU memory overhead relative to standard GRPO. Peak consumption increases by 3.0 GiB for Qwen-7B, 3.0 GiB for Gemma-4B, and 4.9 GiB for Qwen-3B, corresponding to average temporal overheads of just 11.4%, 9.1%, and 37.5%, respectively. Consequently, the memory footprint of AMIR-GRPO remains essentially comparable to that of GRPO.

E Error Taxonomy

We adopt a high-coverage error taxonomy tailored to the systematic diagnosis of failures in automated mathematical problem solving. The framework is intentionally diagnostic: each label corresponds to a distinct failure mode, enabling targeted, actionable remediation. The taxonomy spans the key stages of a solution pipeline—problem formulation, core inference, and answer realization—so that errors can be localized to where they arise.

The taxonomy includes the following categories:

1. *Calculation error.* Incorrect arithmetic or algebraic manipulation despite an otherwise correct setup.
2. *Conceptual error.* Use of an inappropriate theorem, formula, definition, or overall strategy.
3. *Reasoning error.* An invalid logical step, unsupported inference, missing case analysis, or circular argumentation.
4. *Modeling error.* Incorrect formalization (e.g., variable definitions, equations/constraints, representations, or case structure).
5. *Constraint error.* Failure to enforce domain/feasibility constraints or to eliminate extraneous solutions.
6. *Prompt misinterpretation.* Misreading the question or misconstruing given quantities, conditions, or objectives.
7. *Format error.* Correct reasoning but an incorrect final answer presentation, or noncompliance with the required output format.
8. *Other.* Instances not covered above.

F Case Analysis

Problem: AMC23 Q6

Consider the set of complex numbers z satisfying $|1+z+z^2|=4$. The maximum value of the imaginary part of z can be written in the form $\frac{\sqrt{m}}{n}$, where m and n are relatively prime positive integers. What is $m+n$?

Model Output (Qwen-7B$_{\text{GRPO}}$)

To solve this problem, let's start by expressing z as $z = x + yi$, where x and y are real numbers, and i is the imaginary unit. We need to find the maximum value of y such that $|1+z+z^2| = 4$.
First, compute $1+z+z^2$:

$$1+z+z^2 = 1+(x+yi)+(x+yi)^2 = (1+x+x^2-y^2)+(y+2xy)i.$$

The magnitude of this complex number is given by:

$$|1+z+z^2| = \sqrt{(1+x+x^2-y^2)^2+(y+2xy)^2} = 4.$$

We want to maximize y. To do this, we can use symmetry and geometric interpretation. Consider the points on the complex plane where the condition holds. One useful approach is to consider specific values of x and y that might simplify the expression.
Let's try $x = -\frac{1}{2}$. Then:

$$1+z+z^2 = \frac{1}{4}+y^2+\left(y-\frac{1}{2}\right)i \text{ (wrong simplification)} \implies 1+z+z^2 = \frac{3}{4}-y^2$$

So,

$$|1+z+z^2| = \sqrt{\left(\frac{1}{4}+y^2\right)^2+\left(y-\frac{1}{2}\right)^2} = 4.$$

Simplify the equation:

$$\left(\frac{1}{4}+y^2\right)^2+\left(y-\frac{1}{2}\right)^2 = 16.$$

Let $u = y-\frac{1}{2}$, then $y = u+\frac{1}{2}$. Substitute y in the equation:

$$\left(\frac{1}{4}+\left(u+\frac{1}{2}\right)^2\right)^2+u^2 = 16.$$

By trial and error or solving the quadratic, we find that the maximum value of y is $\frac{\sqrt{15}}{2}$. This can be verified by checking the discriminant and solving the quadratic equation.
Thus, the maximum value of the imaginary part of z is $\frac{\sqrt{15}}{2}$. Here, $m = 15$ and $n = 2$, so $m+n = \boxed{17}$.

Model Output (Qwen-7B$_{\text{AMIR-GRPO}}$)

To solve this problem, let's start by expressing z in its general form as $z = x + yi$, where x and y are real numbers, and i is the imaginary unit. We need to find the condition $|1+z+z^2| = 4$.
First, we calculate z^2:

$$z^2 = (x+yi)^2 = x^2-y^2+2xyi.$$

Then, we substitute z and z^2 into the expression $1+z+z^2$:

$$1+z+z^2 = 1+(x+yi)+(x^2-y^2+2xyi) = (1+x+x^2-y^2)+(y+2xy)i.$$

The magnitude of this complex number is given by:

$$|1+z+z^2| = \sqrt{(1+x+x^2-y^2)^2+(y+2xy)^2} = 4.$$

Squaring both sides, we get:

$$(1+x+x^2-y^2)^2+(y+2xy)^2 = 16.$$

To find the maximum value of the imaginary part y, we need to analyze the equation. One effective approach is to consider specific values of x and y that might simplify the equation. Let's try $x = -\frac{1}{2}$. Then the equation becomes:

$$1-\frac{1}{2}+\left(-\frac{1}{2}\right)^2-y^2 = 1-\frac{1}{2}+\frac{1}{4}-y^2 = \frac{3}{4}-y^2.$$

And for the imaginary part:

$$y+2\left(-\frac{1}{2}\right)y = y-y = 0.$$

So the equation simplifies to:

$$\left(\frac{3}{4}-y^2\right)^2 = 16.$$

Taking the square root of both sides, we get:

$$\frac{3}{4}-y^2 = \pm 4.$$

This gives us two cases: 1. $\frac{3}{4}-y^2 = 4$, which is not possible since $\frac{3}{4}-4 = -\frac{13}{4}$. 2. $\frac{3}{4}-y^2 = -4$, which simplifies to:

$$y^2 = \frac{3}{4}+4 = \frac{19}{4},$$

so

$$y = \pm\frac{\sqrt{19}}{2}.$$

The maximum value of y is $\frac{\sqrt{19}}{2}$. Therefore, $m = 19$ and $n = 2$, and $m+n = 21$.
Thus, the answer is: $\boxed{21}$.

Case 1: GRPO: failure vs. AMIR-GRPO: success

Problem: AIME25 Q1

In $\triangle ABC$ points D and E lie on $\overline{AB}$ so that $AD < AE < AB$, while points F and G lie on $\overline{AC}$ so that $AF < AG < AC$. Suppose $AD = 4$, $DE = 16$, $EB = 8$, $AF = 13$, $FG = 52$, and $GC = 26$. Let M be the reflection of D through F, and let N be the reflection of G through E. The area of quadrilateral $DEGF$ is 288. Find the area of heptagon $AFNBCEM$.

Model Output (Qwen-3B$_{\text{GRPO}}$)

To find the area of the heptagon $AFNBCEM$, we need to understand how the reflections and the given areas can help us. First, let's calculate the total area of $\triangle ABC$.
We know the area of quadrilateral $DEGF$ is 288. Since M is the reflection of D through F and N is the reflection of G through E, quadrilaterals $DMEF$ and $ENGF$ will each have the same area as $DEGF$, which is 288. This means the total area added by these reflections is 576.

$$[DMEF] = [ENGF] = [DEGF] = 288 \text{ (reflection misinterpretation)}$$

Now, let's determine the area of $\triangle ABC$. The segments AD, DE, and EB divide AB into three parts: 4, 16, and 8, making $AB = 4 + 16 + 8 = 28$. Similarly, the segments AF, FG, and GC divide AC into three parts: 13, 52, and 26, making $AC = 13 + 52 + 26 = 91$.
Since the area of $DEGF$ is 288 and this quadrilateral is formed by segments of the sides of $\triangle ABC$, we can use the fact that the area of the whole triangle is the sum of the areas of $\triangle ADE$, $\triangle DEG$, $\triangle EFG$, and $\triangle GCF$, plus the area of quadrilateral $DEGF$.
However, a simpler approach is to recognize that the reflections double the area of the quadrilateral $DEGF$ when considering the whole triangle. Therefore, the area of $\triangle ABC$ is twice the area of $\triangle ABC$ minus the area of the reflected quadrilaterals (since they overlap the original quadrilateral).
Thus, the area of $\triangle ABC$ is:

$$\text{Area of } \triangle ABC = \frac{\text{Area of } \triangle ABC - \text{Area of } DEGF}{2} + 2 \times 288 = 2 \times 288 = 576.$$

The area of the heptagon $AFNBCEM$ is the area of $\triangle ABC$ minus the area of the four small triangles that are not part of the heptagon, which is the area of $\triangle ADE + \triangle EGF + \triangle FGC + \triangle DME$.
Since the area of the four small triangles is the same as the area of $\triangle ABC$ minus the area of the heptagon, we have:

$$\text{Area of heptagon } AFNBCEM = 576 - 288 = 288.$$

So, the area of the heptagon $AFNBCEM$ is: $\boxed{576}$.

Model Output (Qwen-3B$_{\text{AMIR-GRPO}}$)

To solve this problem, we start by finding the total area of triangle ABC. We know the area of quadrilateral $DEGF$ is 288. First, let's find the areas of triangles ADE and GEC.
Since $AD = 4$ and $DE = 16$, $AE = AD + DE = 4 + 16 = 20$. Similarly, since $AF = 13$ and $FG = 52$, $AG = AF + FG = 13 + 52 = 65$.
We can use the fact that the area of a triangle is proportional to the base if the height from the opposite vertex is the same. For triangles ADE and GEC, their areas will be proportional to their bases AE and EC respectively.
Since E divides AB into segments $AE = 20$ and $EB = 8$, the ratio $AE : AB = 20 : 28 = 5 : 7$.
Similarly, since F divides AC into segments $AF = 13$ and $FC = 42$, the ratio $AF : AC = 13 : 65 = 1 : 5$. (inconsistent: FC and AC are wrong)

$$\implies \quad AF = 13,\ FG = 52,\ GC = 26; \Rightarrow; FC = 78,\ AC = 91,\ \frac{AF}{AC} = \frac{13}{91}$$

Now, let's denote the area of $\triangle ABC$ as K. The area of $\triangle ADE$ is $\frac{5}{7} \cdot \frac{1}{5} \cdot K = \frac{1}{7}K$. The area of $\triangle GEC$ is $\frac{2}{7} \cdot \frac{4}{5} \cdot K = \frac{8}{35}K$.
The area of quadrilateral $DEGF$ is the area of $\triangle ADE$ plus the area of $\triangle GEC$ plus the area of $\triangle DFE$. Since the area of $\triangle DFE$ is the same as the area of $\triangle DEG$ (due to reflections), we have:

$$\text{Area of } \triangle DFE = \text{Area of } \triangle DEG = \frac{1}{2} \times \text{Area of } \triangle DEF = \frac{1}{2} \times 288 = 144$$

Thus, the area of $\triangle DEF$ is 144. Therefore, the area of $\triangle ADE$ plus the area of $\triangle GEC$ plus the area of $\triangle DFE$ is:

$$\frac{1}{7}K + \frac{8}{35}K + 144 = 288$$

Combining the fractions, we get:

$$\frac{5}{35}K + \frac{8}{35}K + 144 = 288 \implies \frac{13}{35}K + 144 = 288 \implies \frac{13}{35}K = 144 \implies K = 144 \cdot \frac{35}{13} = 360$$

So, the area of $\triangle ABC$ is 360. Now, we need to find the area of the heptagon $AFNBCEM$. This area is the area of $\triangle ABC$ minus the areas of $\triangle ADE$, $\triangle GEC$, and $\triangle DFE$:

$$360 - \left(\frac{1}{7} \cdot 360 + \frac{8}{35} \cdot 360 + 144\right) = 360 - (51.43 + 92.57 + 144) = 360 - 288 = 144$$

However, since the reflections create additional areas, the correct area of the heptagon is:

$$360 - 288 = 144$$

Thus, the area of the heptagon $AFNBCEM$ is $\boxed{432}$."

Case 2: GRPO: failure vs. AMIR-GRPO: failure

The Cost of Incoherent Context Grows with Capability: A Control for Exemplar Retrieval

Srinivas Harish
New York University
srinivas.harish@nyu.edu

Abstract

Comparisons of retrieved and randomly selected exemplars in mathematical in-context learning confound query relevance with changes in the demonstration pool. We separate these effects by assigning each query the demonstrations retrieved for another query. This derangement preserves the aggregate demonstrations, counts, token budget and retrieval-induced selection bias while changing the query-to-exemplar assignment.

On GSM8K, derangement changes accuracy by -0.24 pp (90% CI [-0.85, +0.37]). The pooled effect is equivalent to zero within 1.0 pp across conditions that include embedding retrieval, a gold-computational-graph oracle and a 7B model. On MathQA, the change is -8.36 pp. Near-duplicates occur above cosine 0.95 for 64.6% of MathQA queries and no GSM8K queries. Removing them changes the matching term by -13.56 pp relative to queries unaffected by removal.

Pairing each retrieved problem with another retrieved rationale changes accuracy by -22.40 to -52.24 pp. Across 8 conditions and 3 model families, this change correlates with excess copying of demonstration answers (Spearman -0.976). Across 7 models, the correlation between the accuracy change and unpermuted accuracy is +0.286. These results support reporting pool near-duplicate density alongside retrieval gains and checking problem-rationale alignment during prompt construction, since retrieval metrics remain unchanged under this within-demonstration permutation.

1 Introduction

Exemplar retrieval is used in mathematical in-context learning. Embedding retrievers select demonstrations by semantic similarity to the query (Liu et al., 2022; Rubin et al., 2022); complexity- and diversity-based selectors reshape which demonstrations appear (Fu et al., 2023; Zhang et al., 2023); and structure-aware retrievers match the computational graph of the query to the computational graphs of candidate demonstrations, on the explicit argument that semantic retrieval "prioritise[s] superficial linguistic similarity over deep structural alignment" (Yang et al., 2024b).

Retrieval changes both the assignment of demonstrations to queries and the frequency with which demonstrations are selected from the pool. Comparisons against random selection vary both, leaving their contributions to accuracy unresolved.

Let $R_r(q)$ be the k demonstrations retriever r selects for query q and σ a derangement of the test set. Assigning $R_r(\sigma(q))$ to q preserves the empirical distribution of retrieved sets, including their counts, aggregate token budget and concentration, while randomising the query assignment. Comparing this condition with the original retrieval condition estimates the contribution of matching at a fixed demonstration distribution. Resampling retrieved sets preserves that distribution in expectation; a derangement also prevents queries from retaining their original set, subject to the collision handling in §2.2.

Query assignment and near-duplicates. On GSM8K, deranging the query-to-exemplar assignment changes accuracy by -0.24 pp (90% CI [-0.85, +0.37], TOST $p = 0.0186$). The pooled effect meets the pre-stated equivalence margin of 1.0 pp. Across three models, including a 7B model, no condition changes by more than 2.01 pp and no McNemar p is below 0.122. For the oracle retriever, realised structural similarity falls from 0.941 to 0.425, near the uniform-random level. On MathQA, the same derangement changes accuracy by -8.36 pp ([-10.34, -6.38]); the matching term accounts for the retrieval gain ($\Delta_{\text{total}} = +10.20$, $\Delta_{\text{match}} = +12.60$, $\Delta_{\text{pool}} = -1.65$ pp).

No GSM8K test query has a pool neighbour at cosine 0.95 or above, compared with 64.6% on

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 86–100
October 29, 2026 ©2026 Association for Computational Linguistics

MathQA, where 26.4% exceed 0.99 and the median nearest-neighbour similarity is 0.973. Such neighbours can share a problem template with different numerals, providing a worked solution method for the query. Within MathQA, correct assignment improves accuracy by +18.11 pp for queries with a near-duplicate and +2.54 pp for those without one. Removing near-duplicates from each query's candidate list changes the matching term by −13.56 pp relative to unaffected queries.

Fu et al. (2023) previously described MathQA training and development questions with similar wording and small changes to the numerals, reporting retrieval−random gains of +3.5 pp on GSM8K and +36.5 pp on MathQA. Jie and Lu (2023) reproduce this asymmetry and find similar performance for random and least-similar selection on MathQA. The dataset papers also describe differences in construction (Amini et al., 2019; Cobbe et al., 2021). We test the near-duplicate account by isolating assignment from pool composition, measuring near-duplicate availability and manipulating it within MathQA.

Within-demonstration permutations. Permuting problem-rationale pairings reduces accuracy in every math-tuned condition on both corpora, with or without near-duplicates (Table 1). Presentation order, included as a positive control based on prior work (Lu et al., 2022), has no significant effect in any of the four conditions. The general base model has an INTRA change of −2.16 pp, while the least disruptive step permutation, STEPTOPO, changes accuracy by −0.99 pp ($p = 0.203$). In all four math-tuned 1.5B conditions, permuting problem-rationale pairings has a larger effect than permuting query assignments. On MathQA, the respective changes are −25.55 and −11.85 pp; on GSM8K, the assignment effect is near zero, so a ratio would be unstable. For the math-specialised and general 1.5B models, within-set derangement changes accuracy by −22.40 and −2.16 pp, respectively, a factor of 10.4 in magnitude.

Scope. Prior work establishes the importance of internal correspondence (Wang et al., 2023; Madaan and Yazdanbakhsh, 2022) and reports retrieval gains on MathQA associated with near-duplicate examples. Our contribution is a control that separates assignment from pool composition and measures their respective effects.

Contributions.

1. A derangement control for exemplar retrieval and an exact three-term decomposition of any reported gain into pool-reshaping, difficulty-coupling and query-matching terms (§2).
2. A test of the near-duplicate account of mathematical retrieval gains, which the literature states as an observation (Fu et al., 2023; Jie and Lu, 2023). We quantify pool near-duplicate density by cosine, delete near-duplicates from each query's candidate list, and estimate the effect as a difference-in-differences against an internal placebo (§4).
3. The permutation hierarchy inside the demonstration, placing the internal and external senses of relevance on one commensurable axis, with the manipulations of Wang et al. (2023) and Madaan and Yazdanbakhsh (2022) recovered as token-preserving permutations under one protocol (§5).
4. A comparison across 8 conditions and 3 model families relating accuracy changes to excess answer copying (Spearman −0.976) and unpermuted accuracy (+0.286). A pre-registered control that restores correct problem-answer pairs produces a larger GSM8K loss, indicating that answer-copying alone does not explain it.
5. Recommendations for reporting near-duplicate density and checking problem-rationale alignment during prompt construction (§6).

The released artifacts include per-item generations and CPU scripts for recomputing the reported statistics. GPU computation is required for generation.

2 Permutation as a Measurement Instrument

2.1 The decomposition

Fix a pool P, shot count k, and retriever r mapping a query to an ordered $R_r(q) \subset P$, $|R_r(q)| = k$; write $\mathrm{Acc}[\cdot]$ for accuracy under a prompt-construction rule and U_k for a uniform draw of k from P. The literature reports $\Delta_{\text{total}}(r) = \mathrm{Acc}[R_r(q)] - \mathrm{Acc}[U_k]$. Let σ be a derangement of the test indices ($\sigma(i) \neq i$ everywhere) and σ_s one confined to strata of a nuisance variable s, here the gold step count. Then, writing $A_\pi =$

$\mathrm{Acc}[R_r(\pi(q))]$,

$$\underbrace{A_\sigma - \mathrm{Acc}[U_k]}_{\Delta_{\text{pool}}} + \underbrace{A_{\sigma_s} - A_\sigma}_{\Delta_{\text{diff}}} + \underbrace{A_{\text{id}} - A_{\sigma_s}}_{\Delta_{\text{match}}} = \Delta_{\text{total}}. \quad (1)$$

The terms telescope, so Equation 1 requires no modelling assumption. Δ_{pool} measures the effect of changing the distribution of demonstrations shown; Δ_{diff} measures the effect of coupling demonstration and query difficulty through the specified strata; and Δ_{match} measures the remaining query-specific matching effect. An unrestricted derangement changes both difficulty coupling and query matching, which motivates the stratified condition. U_k samples uniformly from the pool, whereas $R_r(\sigma(q))$ uses the empirical distribution of retrieved sets, which can be highly concentrated (Appendix J). Comparisons of retrieval against random selection estimate the sum of these terms.

2.2 The experimental conditions

Write a demonstration as a triple (x_j, y_j, a_j): problem, rationale, stated final answer. We call each experimental condition an *arm*: one complete evaluation of the model over the test set, differing from the others only in the permutation applied to the retrieved demonstrations. The unpermuted prompt for query q is the concatenation of $(x_j, y_j, a_j)_{j=1}^{k}$ for j indexing $R_r(q)$, most-similar demonstration placed adjacent to q, followed by q.

Outside the demonstration. DER replaces $R_r(q)$ by $R_r(\sigma(q))$ for a derangement σ over test indices. Because a collapsed top-k can return the same set to different queries, we derange over sets rather than over indices and repair any position handed its own set back, so no item sees its own treatment. DERSTRAT is the same with σ confined to strata of the gold step count $\min(n_{\text{steps}}, 6)$; singleton strata are left in place and counted without merging strata, and the within-stratum collision repair fails on $18/1319$ GSM8K STRUCT items, which do see their own set. ORDER applies a derangement over the k slots, permuting the presentation order of the selected demonstrations.

Inside the demonstration. All three hold $R_r(q)$ fixed. INTRA applies a derangement π over the k slots and prints $(x_j, y_{\pi(j)}, a_{\pi(j)})$. The multiset of problems, the multiset of rationales, the multiset of answers, the token count, the retrieval similarity and the pool composition are identical to the unpermuted arm for each item, and every rationale shown remains a correct worked solution to a problem that is itself in the prompt. INTRACORRECT prints $(x_j, y_{\pi(j)}, a_j)$ with the same π from the same seed, so the stated answer is correct for the displayed problem. STEPORDER permutes the lines of each y_j; STEPTOPO and STEPVIOL do so respecting and violating the derivation DAG read off the arithmetic, applied to a rationale in both arms or neither so the two are matched. NUMPERM deranges the numeric literals within y_j, preserving the equation skeleton, the entities and the token count while destroying the arithmetic, the invalid-reasoning manipulation of Wang et al. (2023) realised as a permutation under the same evaluation protocol.

The assignment and within-set permutation controls preserve the aggregate prompt token multiset; within-set permutations preserve it per item. The INTRACORRECT control also changes which final answer accompanies each deranged rationale. Appendix A gives the arm table.

3 Experimental Setup

Corpora and graphs. For GSM8K (Cobbe et al., 2021), the 7473-problem training split forms the pool and all 1319 test problems are queried. MathQA (Amini et al., 2019) supplies a 8000-problem pool and 1000 test problems. Both are in the evaluation suite of Yang et al. (2024b), whose credited mechanism motivates this study. GSM8K solutions carry inline `<<expr=val>>` annotations, from which we recover per step the operator sequence, operand count, and which earlier results the step consumes; MathQA ships the graph in its `linear_formula` field, whose #k references give the dependency edges. Graph similarity combines longest-common-subsequence agreement over per-step operator tuples, depth agreement and dependency-pattern agreement.

Retrievers. SEM ranks by cosine similarity of BGE-base-en-v1.5 embeddings (Xiao et al., 2024) of the problem statements, the retriever family used by Yang et al. (2024b). STRUCT ranks by computational-graph similarity to the query's gold graph, ties broken by fixed random priority rather than pool order. STRUCT is an oracle and is not deployable; it provides a control with verified structural matching before derangement. It places a demonstration carrying the query's exact gold operator sequence into 69.3% of slots (against 0.8%

for the embedding retriever and 0.7% for a uniform draw) and its gold step count into 98.7% (against 22.2% and 21.2%). On GSM8K, this oracle improves accuracy by about one point.

Models and decoding. Qwen2.5-Math-1.5B, Qwen2.5-Math-7B (Yang et al., 2024a) and, as a same-size non-math control, Qwen2.5-1.5B; all three are base models evaluated few-shot. §5 adds four checkpoints to separate capability from task specialisation: Qwen2.5-0.5B and Qwen2.5-3B, OLMo-2-1B (Team OLMo et al., 2025), all base, and Phi-3.5-mini-instruct (Abdin et al., 2024). Generation used one 32 GB GPU and greedy decoding, so RAND and TRUE are deterministic and the only randomness is the derangement draw, whose seeds are literals in the released `run_arms.py` (12345 for RAND; 1000, 2000, 3000, 4000 plus the replicate index for the others). INTRACORRECT draws INTRA's permutation from the same seed, matching the two arms item by item. $k = 4$, most-similar demonstration adjacent to the query. All arms answer the same queries, so every contrast is paired. We use exact-binomial McNemar tests per replicate, report the largest replicate p-value, compute paired bootstrap intervals over test items and apply Holm correction across each hierarchy. The random baseline is averaged over independent draws. Appendix A gives full arm definitions and a real prompt fragment under INTRA (Figure 1).

4 The Assignment Term and Near-Duplicate Density

Table 1 reports accuracy under each permutation for both corpora.

GSM8K assignment effects. Retrieval gains over random selection are −0.53 and +2.12 pp for the math 1.5B model, −0.30 and +0.91 pp for the general 1.5B model, and −0.08 and +1.14 pp for the math 7B model. None is significant. Our prediction that the general model would exceed +3 pp, based on less specialised pretraining, was not supported.

Derangement changes accuracy by −0.24 pp over 1319 paired items (90% CI [−0.85, +0.37]). A TOST equivalence test meets the pre-stated 1.0 pp margin ($p = 0.0186$). The per-condition minimum detectable effect is 2.88 pp at 80% power, so individual conditions cannot exclude a four-point effect; the pooled analysis, clustered by item, excludes a one-point effect.

At 7B, random selection reaches 84.31 accuracy, and derangement changes accuracy by −1.02 pp ($p = 0.364$) and −0.38 pp ($p = 0.878$), with an MDE of 2.54 pp. Related results vary by model size. Table 2 of Yang et al. (2024b) reports +1.3 pp at 70B, +0.1 at 1–1.5B and performance below random at 8B. Fu et al. (2023) report +3.5 pp on GSM8K for complexity-based selection. Our embedding and gold-graph retrievers show no significant gain at either evaluated scale.

MathQA assignment effects. Retrieval improves accuracy by $\Delta_{\text{total}} = +10.20$ pp for SEM and +4.10 pp for STRUCT, from a random baseline of 58.20. Derangement changes accuracy by −11.85 pp ($p = 6.9 \cdot 10^{-8}$) and −3.60 pp ($p = 0.064$), respectively. The pooled change is −8.36 pp ([−10.34, −6.38]), compared with −0.24 on GSM8K. Equation 1 gives matching terms of +12.60 and +5.40 pp, with pool terms of −1.65 and +0.50 pp. The matching terms account for the retrieval gains, and the encoder comparison in Appendix E gives equal matching terms to two decimal places. On GSM8K, the corresponding terms are +0.00 and +1.40 pp (Table 10).

Pool near-duplicate density. On GSM8K, the mean nearest-neighbour cosine is 0.807, and 0.0% of 1319 queries have a pool neighbour at 0.95 or above. On MathQA, the mean is 0.947, the median is 0.973, and 64.6% and 26.4% of queries exceed 0.95 and 0.99, respectively. For example, the MathQA query "sari and ken climb up a mountain … sari wakes up at 07:00" has a nearest pool problem with cosine 0.9997 that instead specifies "… sari wakes up at 05:00". The gold answers differ, but the neighbour supplies a solution method for the same template.

Despite its low lexical near-duplicate density, GSM8K has a pool problem with the same gold operator signature for 78.9% of queries, with a median of 29 candidates. This structural availability produces no significant retrieval gain in the oracle condition.

Near-duplicate removal. The corpora differ in format, answer type, rationale style and difficulty, so we also manipulate near-duplicate availability within MathQA. For every MathQA query we delete from its candidate list every pool item at cosine $\geq \tau$ of it, for a threshold $\tau = 0.95$, then re-rank. The pool, retriever, model, k, prompt format and test set are unchanged; the candidate lists

corpus	arm	SEM		STRUCT	
		acc (%)	Δ (pp)	acc (%)	Δ (pp)
GSM8K, general 1.5B	uniform random	66.19			
	retrieved (unpermuted)	65.88	-0.30	67.10	+0.91
	DER: query↔set deranged	66.49	+0.61	66.79	-0.30
	DERSTRAT: deranged within difficulty stratum	66.26	+0.38	67.02	-0.08
	INTRA: problem↔rationale deranged in-set	63.72	-2.16	64.97	-2.12
GSM8K, math 1.5B	uniform random	74.45			
	retrieved (unpermuted)	73.92	-0.53	76.57	+2.12
	DER: query↔set deranged	75.32	+1.40	74.56	-2.01
	DERSTRAT: deranged within difficulty stratum	73.92	+0.00	75.17	-1.40
	retrieved (unpermuted), finer-scale stage	73.77		76.80	
	ORDER: presentation order permuted	74.53	+0.76	76.50	-0.30
	STEPTOPO: steps reordered within the derivation DAG	72.78	-0.99	74.75	-2.05
	STEPVIOL: steps reordered against the derivation DAG	71.34	-2.43	71.42	-5.38
	STEPORDER: rationale step order permuted	65.05	-8.72	66.26	-10.54
	NUMPERM: rationale numbers deranged	55.19	-18.57	58.91	-17.89
	INTRA: problem↔rationale deranged in-set	51.52	-22.40	31.31	-45.26
GSM8K, math 7B	uniform random	84.31			
	retrieved (unpermuted)	84.23	-0.08	85.44	+1.14
	DER: query↔set deranged	83.21	-1.02	85.06	-0.38
	DERSTRAT: deranged within difficulty stratum	84.00	-0.23	85.63	+0.19
	INTRA: problem↔rationale deranged in-set	31.99	-52.24	28.28	-57.16
MathQA, math 1.5B	uniform random	58.20			
	retrieved (unpermuted)	68.40	+10.20	62.30	+4.10
	DER: query↔set deranged	56.55	-11.85	58.70	-3.60
	DERSTRAT: deranged within difficulty stratum	55.80	-12.60	56.90	-5.40
	retrieved (unpermuted), finer-scale stage	68.80		63.60	
	ORDER: presentation order permuted	66.30	-2.50	62.70	-0.90
	NUMPERM: rationale numbers deranged	38.40	-30.40	44.00	-19.60
	INTRA: problem↔rationale deranged in-set	42.85	-25.55	46.05	-16.25

Table 1: The permutation hierarchy. Δ is against the unpermuted retrieved arm of the same retriever, except the unpermuted row itself, which is against uniform random selection. Rows above the inner rule permute something outside a demonstration, rows below permute something inside one. All arms answer the same queries; the permutation controls preserve the aggregate token multiset. Accuracies for the finer scales come from a separate generation stage with its own RAND draw, so they are compared against that stage's unpermuted arm. Exact McNemar and Holm-adjusted p-values, and bootstrap intervals, in Table 4.

exclude near-duplicates. After pruning, no top-4 set contains a near-duplicate, and mean maximum query-demonstration cosine falls from 0.947 to 0.887. We registered the prediction that Δ_{match} would fall from +12.60 pp to roughly $+2.5$ pp.

Δ_{match} falls from +12.60 to +5.25 pp ([+2.10, +8.45]) for SEM and from +5.40 to -0.15 pp ([-3.15, +2.95]) for STRUCT; the registered prediction was consistent with STRUCT, whose remaining effect is not significant, and below the observed SEM effect, which remains significant at +5.25 pp.

Generation drift and the pruning estimate. Greedy decoding with identical prompts produces different outputs across generation stages. The two stages' RAND arms have byte-identical demonstration sets on $1000/1000$ items, yet accuracy differs by up to 1.00 pp, with 6.0% of predictions changing. Batch composition changes the kernels' reduction order, so we use a difference-in-differences estimate to account for drift between stages.

The 354 queries with no pool neighbour above τ form the placebo group. Their top-4 sets are verified identical across stages for both retrievers. The matching-term change is +1.41 pp in this group and -12.15 pp for the 646 treated queries, giving a difference-in-differences estimate of -13.56 pp (95% CI [-19.60, -7.33], $p = 1.0 \cdot 10^{-4}$) for SEM and -9.90 pp ([-15.30, -4.54], $p = 6.0 \cdot 10^{-4}$) for STRUCT. This adjustment removes the drift measured in the placebo group.

Pruning leaves residual similarity of 0.887, compared with GSM8K's 0.807, and also changes candidate quality, making $\Delta_{\text{pool}} = -2.10$ and $\Delta_{\text{diff}} = -3.15$ pp negative. Removing near-duplicates reduces most of the assignment benefit, but the pooled change remains -1.61 pp ([-3.40, +0.15]), which is not equivalent to zero. Within-demonstration permutation still changes accuracy by -12.90 pp ($p = 2.0 \cdot 10^{-11}$) and -8.80 pp ($p = 5.6 \cdot 10^{-6}$). These effects are about half their unpruned magnitudes, from a baseline 9.3 points lower, and establish that within-demonstration sensitivity persists after pruning.

In the observational comparison, correct assignment improves accuracy by +18.11 pp for queries whose nearest pool neighbour exceeds cosine 0.95 ($n = 646$), and by +2.54 pp for those below the threshold ($n = 354$). The difference is approximately sevenfold under the same retriever, model and prompt format. The fitted slope against realised similarity is +50.8 pp per unit cosine on MathQA and −5.7 on GSM8K, although the MathQA effect is not monotone across quartiles (Appendix G).

Similarity and nuisance checks. For STRUCT on GSM8K, realised graph similarity falls from 0.941 unpermuted to 0.599 under stratified derangement and 0.425 under unrestricted derangement, compared with 0.418 for uniform selection (Appendix C). Accuracy changes by −2.01 pp (95% CI [−4.09, −0.04], McNemar $p = 0.122$). The INTRA arm matches its unpermuted counterpart to four decimals on token budget, step count, embedding similarity and graph similarity, excluding changes in these quantities as explanations for its accuracy loss.

5 Within-Demonstration Effects

Table 1 compares permutations between and within demonstrations. Deranging the problem-to-rationale correspondence changes accuracy by −22.40 pp ($p = 5.5 \cdot 10^{-49}$) on GSM8K and −25.55 pp on MathQA, compared with −0.24 and −8.36 for deranging the query-to-set assignment. The prompt in the INTRA arm contains k correct worked solutions to k retrieved problems; each solution remains valid for its source problem but is displayed beneath another problem.

Internal and external relevance. Wang et al. (2023) define relevance within a demonstration, where each rationale component corresponds to part of its problem. Retrieval studies instead measure relevance to the test query. We vary both under the same protocol.

Under DER, each rationale remains paired with its original problem, but the set is reassigned to a different query. Under INTRA, the set remains the one retrieved for the query, but its problems and rationales are paired incorrectly. Both conditions preserve individually valid rationales and the aggregate multisets of problems and rationales; INTRA also preserves those multisets for each query. On GSM8K, the respective accuracy changes are −0.24 and −22.40 pp; on MathQA, they are −8.36 and −25.55 pp.

Wang et al. (2023) report GSM8K accuracy decreasing from 48.5 to 27.5 and 11.0 when internal relevance is removed, compared with 39.5 under invalid reasoning. Our NUMPERM (−18.57 pp) and STEPORDER (−8.72 pp) conditions reproduce this ordering. Their no-relevance conditions also change the stated answer. Madaan and Yazdanbakhsh (2022) keep that answer correct in a RANDOM TEXT condition that substitutes a valid rationale for another problem, obtaining 2.98% accuracy on GSM8K. Our INTRACORRECT condition performs this substitution within the prompt. Comparing it with INTRA measures the additional effect of correcting the stated answer while holding the deranged rationale fixed.

Model size and specialisation. On GSM8K, within-set derangement changes accuracy by −2.16 pp ($p = 0.258$) for general 1.5B, −22.40 pp for math-tuned 1.5B and −52.24 pp ($p = 5.7 \cdot 10^{-172}$) for math-tuned 7B. The 7B model achieves 84.31 with random demonstrations and 31.99 under INTRA. None of these models has a significant query-assignment effect.

Four further models distinguish size from task specialisation. OLMo-2-1B changes by −9.60 pp from 32.80 accuracy ($p = 0.0022$, $n = 250$). Phi-3.5-mini changes by −4.00 pp from 78.00, which is not significant ($p = 0.45$, $n = 100$). The general Qwen 0.5B, 1.5B and 3B models have unpermuted accuracies of 36.54, 65.88 and 75.82, with changes of −1.06, −2.16 and +0.23 pp, none significant. General Qwen 3B and Qwen2.5-Math-1.5B have similar unpermuted accuracies of 75.82 and 73.92, but changes of +0.23 and −22.40 pp, respectively. Across 7 models from 3 families, the Spearman correlation between unpermuted accuracy and the signed INTRA change is +0.286.

Answer copying. Under INTRA, 100.0% of outputs contain a well-formed answer, and generation lengths remain similar to those in the unpermuted condition (Appendix K). We measure copying as the fraction of predictions equal to a printed demonstration answer. Excess copying relative to the unpermuted condition correlates with the signed accuracy change across 8 conditions and 3 families (Spearman −0.976, Table 3).

OLMo-2-1B's copy rate increases from 5.2% to 8.8%, and Phi-3.5-mini's from 5.0% to 6.0%. Their excess rates are +3.6 and +1.0 pp, compared with 43.1 pp for math-tuned 7B. Each excess is

measured against the model's own unpermuted rate. For math-tuned 1.5B, that rate is 6.4%, and the gold answer occurs among the demonstration answers on 6.1% of items, so a matching answer does not necessarily indicate erroneous copying.

Copying occurs in 46.0% and 69.6% of the math-tuned 1.5B model's lost items. The non-copying subsets still lose −14.66 and −24.28 pp, with a loss of −26.47 at 7B. These associations are consistent with greater reliance on demonstrated patterns in the math-tuned models, including when those patterns are inconsistent.

Corrected final answers. INTRACORRECT uses the same rationale derangement and seed as INTRA, but restores the correct final answer for each displayed problem. The two conditions are matched item by item. We pre-registered that an effect near zero under INTRACORRECT would support answer-copying as the explanation and require reinterpreting the result as sensitivity to distractor answers.

On GSM8K, INTRACORRECT changes accuracy by −28.54 pp compared with −22.40 for SEM, and by −52.05 compared with −45.26 for STRUCT. The losses increase by roughly six points. Under INTRA, the rationale derives the displayed answer for another problem; under INTRACORRECT, it also conflicts with the displayed answer. This additional inconsistency accounts for the difference between conditions and indicates sensitivity to the relation between rationale and answer. On MathQA, INTRACORRECT is marginally less costly (−23.05 compared with −25.55). Five-way multiple choice may contribute to this difference, since a copied option letter has a one-in-five chance of being correct under uniform guessing.

Derivation dependencies. On the 38.8% of pool rationales that are not linear chains, permuted both ways so the arms are matched, reordering within the dependency structure changes accuracy by −0.99 pp and violating it by −2.43 pp. The arms also differ in step displacement, preventing attribution of the gap to graph structure (Appendix I).

Presentation order. We included a presentation-order condition as a calibration based on established prompt-order sensitivity (Lu et al., 2022). None of its four effects is significant. The changes are +0.76 pp ($p = 0.480$) and −0.30 pp ($p = 0.817$) on GSM8K, and −2.50 pp ($p = 0.077$) and −0.90 pp ($p = 0.567$) on MathQA. The same items show significant step-reordering effects of −2.43 pp ($p = 0.008$) and −2.05 pp ($p = 0.011$). These comparisons demonstrate sensitivity to some permutations but do not establish presentation-order equivalence. The query-assignment analysis uses its separate equivalence test and per-condition MDE (Appendix H).

6 Practical Implications

Pool measurement. Near-duplicate density can be computed with one embedding pass and no language-model generation. Reporting the fraction of queries above a specified nearest-neighbour similarity threshold helps characterise the pool used in a retrieval evaluation. At $\tau = 0.90$, the fractions are 2.4% on GSM8K and 79.8% on MathQA. The retrieval gains observed on MathQA do not reproduce on GSM8K (Appendix G).

Problem-rationale alignment. Prompt builders should verify that each rationale remains paired with its source problem. Misalignment produces accuracy changes of −22.40 to −45.26 pp in the math-tuned 1.5B conditions while leaving the retrieved demonstrations unchanged. Recall@k, MRR and mean query-demonstration similarity cannot detect this error because they depend on the retrieved set. Table 5 confirms that INTRA matches the unpermuted condition to four decimals on every similarity and nuisance measure.

7 Related Work

Demonstration selection. Similarity-based selection (Liu et al., 2022), learned retrieval (Rubin et al., 2022), complexity-based selection (Fu et al., 2023), clustering-based construction (Zhang et al., 2023) and computational-graph retrieval (Yang et al., 2024b) all report gains against random selection. Peng et al. (2024) find similarity-based selection does not reliably beat random and Zhang et al. (2023) report a GSM8K null. Near-duplicate availability reconciles these with the large MathQA gains of Fu et al. (2023) and Jie and Lu (2023), both of whom attribute the asymmetry to near-duplication; we measure and manipulate it. Its open-domain-QA counterpart is Lewis et al. (2021).

Demonstration content. Min et al. (2022) showed random labels barely hurt classification in-context learning; later work qualified this (Yoo et al., 2022; Kossen et al., 2024), and Pan et al.

(2023) separated task recognition from task learning. Wang et al. (2023) study relevance within demonstrations. We compare this with relevance to the test query under the same protocol.

Relevance controls. Liu et al. (2022) evaluate a farthest-neighbour condition that reduces accuracy by 15 points on NaturalQuestions, where a retrieved neighbour frequently contains the answer. This condition also changes the demonstration distribution. MathQA near-duplicates similarly provide query-specific information through shared solution templates. Prompt-order permutations (Lu et al., 2022) change positions within a selected set. Neither control isolates query assignment at a fixed retrieved-set distribution.

Mathematical NLP. Related work studies retrieval over mathematical content (Ferreira and Freitas, 2020) and permutations of semantic assignments (Williamson et al., 2025; Dias-Alexiou et al., 2025). Our control permutes the assignment of retrieved demonstration sets to queries.

8 Conclusion

Derangement separates the effect of query-specific matching from changes in the retrieved demonstration distribution. The pooled GSM8K assignment effect meets the pre-stated equivalence margin; on MathQA, matching accounts for the retrieval gain, which decreases after near-duplicate removal. Within-demonstration misalignment reduces accuracy on both corpora, with larger losses in math-tuned models and a strong association with excess copying of demonstration answers.

Limitations

The measurements run from 0.5B to 7B, greedy-decoded, at $k = 4$, on two English math-word-problem corpora. The INTRA result covers 7 models from 3 families, but OLMo-2-1B and Phi-3.5-mini are run on subsamples of 250 and 100 items against the 1319 of the others, and Phi-3.5-mini is instruction-tuned where the rest are base checkpoints, so its non-significant result is limited to that evaluation design. The assignment result itself is measured on the original three models only. On GSM8K the absolute retrieval gain is one to two points, so the decomposition there partitions a small quantity. The equivalence test bounds the assignment effect within its stated margin; MathQA provides a comparison with larger retrieval gains. The near-duplicate account rests on one pruning experiment that is partial and also changes candidate quality (residual similarity 0.887 against GSM8K's 0.807, and it turns Δ_{pool} and Δ_{diff} negative); the density it manipulates is measured under 3 embedders and 6 thresholds and across 6 corpora, but the manipulation itself was run on MathQA alone. The permutations are those the released annotations can express, so a demonstration property with no annotation cannot be deranged, the topology-split step permutation is available only for the 38.8% of rationales whose derivation is not a linear chain, and the intra-demonstration result describes systems that construct the problem-rationale pairing, such as self-generated exemplars, clustered pseudo-rationales (Zhang et al., 2023) and distillation corpora, rather than systems that inherit it intact from a corpus.

References

Marah Abdin et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Carolina Dias-Alexiou, Edison Marrese-Taylor, and Yutaka Matsuo. 2025. An in-depth human study of the mathematical reasoning abilities in large language models. In *Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025)*, pages 186–194, Suzhou, China. Association for Computational Linguistics.

Deborah Ferreira and André Freitas. 2020. Natural language premise selection: Finding supporting statements for mathematical text. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2175–2182, Marseille, France. European Language Resources Association.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning. In *International Conference on Learning Representations*.

Zhanming Jie and Wei Lu. 2023. Leveraging training data in few-shot prompting for numerical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10518–10526.

Jannik Kossen, Yarin Gal, and Tom Rainforth. 2024. In-context learning learns label relationships but is not conventional learning. In *International Conference on Learning Representations*. ArXiv:2307.12375.

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2021. Question and answer test-train overlap in open-domain question answering datasets. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1000–1008.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Aman Madaan and Amir Yazdanbakhsh. 2022. Text and patterns: For effective chain of thought, it takes two to tango. *arXiv preprint arXiv:2209.07686*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations:

What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In *Findings of the Association for Computational Linguistics: ACL 2023*. ArXiv:2305.09731.

Keqin Peng, Liang Ding, Yancheng Yuan, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. 2024. Revisiting demonstration selection strategies in in-context learning. *arXiv preprint arXiv:2401.12087*.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, et al. 2025. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2717–2739, Toronto, Canada. Association for Computational Linguistics.

Dane A Williamson, Yangfeng Ji, and Matthew B. Dwyer. 2025. Syntactic blind spots: How misalignment leads to LLMs' mathematical errors. In *Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025)*, pages 1–14, Suzhou, China. Association for Computational Linguistics.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024a. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Xiaocong Yang, Jiacheng Lin, Ziqi Wang, and Chengxiang Zhai. 2024b. Learning by analogy: Enhancing few-shot prompting for math word problem solving with computational graph-based retrieval. *arXiv preprint arXiv:2411.16454*.

Kang Min Yoo, Junyeob Kim, Hyuhng Joon Kim, Hyunsoo Cho, Hwiyeol Jo, Sang-Woo Lee, Sang-goo Lee, and Taeuk Kim. 2022. Ground-truth labels matter: A deeper look into input-label demonstrations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. ArXiv:2205.12685.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *International Conference on Learning Representations*.

A Definitions of the Experimental Conditions

Table 2 defines every arm, and Figure 1 shows a real prompt fragment under INTRA beside its unpermuted counterpart.

arm	demonstrations shown for query q
`rand`	k drawn uniformly from the pool
`r_true`	top-k retrieved for q, most similar adjacent to q
`r_der`	top-k retrieved for $\sigma(q)$
`r_derstrat`	top-k retrieved for $\sigma_s(q)$, σ_s confined to q's step-count stratum
`r_order`	top-k for q, presented in permuted order
`r_intra`	top-k for q; demonstration j's question shown with demonstration $\pi(j)$'s rationale and answer
`r_steporder`	top-k for q; each rationale's own step order permuted
`r_numperm`	top-k for q; numeric literals inside each rationale deranged

Table 2: Arm definitions. σ is a derangement of the test set, π a derangement of the k demonstration slots.

B The Permutation Hierarchy with Exact p-values

Table 4 gives every arm of the GSM8K hierarchy for the math-tuned model under both retrievers, with the paired bootstrap interval, the exact-binomial McNemar p and the Holm-adjusted p, supplementing the accuracies in Table 1.

C Full Similarity and Nuisance Audit

Table 5 is the compact audit quoted in §4 and Table 6 the same audit over every arm. Both report, per arm, the mean prompt tokens and gold steps per demonstration and the mean query-demonstration embedding and graph similarity, confirming that the INTRA arm is matched to its unpermuted arm on all four.

UNPERMUTED (top-k retrieved for the query)

Question: In a basketball game, Cyrus made exactly eighty percent of the shots he attempted. He attempted twenty shots. How…
Answer: Cyrus made 20 x 80/100 = 16 shots. So, he missed 20 - 16 = 4 shots. The answer is 4.
Question: It's payday but Jebb has to pay 10% for the tax. If his pay is $650, how much is his take-home pay?
Answer: Jebb has to pay $650 x 10/100 = $65. Therefore, Jebb's take-home pay is $650 - $65 = $585. The answer is 585.

INTRA (same demonstrations; rationales exchanged)

Question: In a basketball game, Cyrus made exactly eighty percent of the shots he attempted. He attempted twenty shots. How…
Answer: Jebb has to pay $650 x 10/100 = $65. Therefore, Jebb's take-home pay is $650 - $65 = $585. The answer is 585.
Question: It's payday but Jebb has to pay 10% for the tax. If his pay is $650, how much is his take-home pay?
Answer: Cyrus made 20 x 80/100 = 16 shots. So, he missed 20 - 16 = 4 shots. The answer is 4.

Figure 1: Prompt fragment from the released artifacts, showing two of four demonstrations. Both blocks contain the same problems, rationales, answers and tokens. Under INTRA, each rationale appears beneath a different problem while remaining a valid solution to its source problem.

condition	n	Δ accuracy	copy excess
Qwen2.5-Math-7B, sem	1,319	-52.24	$+43.1$
Qwen2.5-Math-1.5B, struct	1,319	-45.26	$+37.4$
Qwen2.5-Math-1.5B, sem	1,319	-22.40	$+11.1$
OLMo-2-1B, sem	250	-9.60	$+3.6$
Phi-3.5-mini, sem	100	-4.00	$+1.0$
Qwen2.5-1.5B general, sem	1,319	-2.16	$+0.0$
Qwen2.5-0.5B general, sem	1,319	-1.06	$+0.0$
Qwen2.5-3B general, sem	1,319	$+0.23$	$+0.0$

Table 3: Signed accuracy change under INTRA and excess answer copying across 8 conditions and 3 model families on GSM8K. Both columns are in percentage points; copy excess is the copy rate minus its unpermuted value. Their Spearman correlation is -0.976. Across 7 models, the correlation between unpermuted accuracy and the signed change is +0.286. The general Qwen checkpoints span 36.54–75.82 accuracy with zero copy excess.

D Near-Duplicate Density Under Three Embedders

The near-duplicate measurement of §4 uses one embedder at one threshold. Table 7 repeats it with 3 embedders over 6 thresholds on both corpora. GSM8K has no near-duplicate under any of them at any threshold from 0.95 up, and MiniLM-L6, which shares neither training nor dimension with the retriever of §3, finds 1.0% of GSM8K queries above 0.85 against bge-base's 21.8%. On MathQA the density is 64.6% for bge-small and 44.5% for MiniLM-L6 at $\tau = 0.95$, against 64.6% for bge-base.

In all 18 embedder-threshold combinations, queries with a pool neighbour above τ gain more from correct assignment than queries without one. For each embedder, the gap increases with τ. At $\tau = 0.99$, the gains are +26.1 pp and +3.9 pp for bge-base, and +26.4 pp and +3.9 pp for MiniLM-L6. Agreement with bge-base on the $\tau = 0.95$ flag has Jaccard similarity 0.903 for bge-small and 0.684 for MiniLM-L6.

Additional corpora. Density is a property of a (pool, queries) pair and needs no model, so it can be taken wherever a training split and a test split exist. Table 8 applies one embedder and one procedure to 6 math word-problem corpora. SVAMP has 54.7% of its queries above cosine 0.99 and 64.3% above 0.95, more than MathQA; ASDiv and AQuA have densities of 5.7% and 3.1% at 0.95; MAWPS and GSM8K have densities of 0.0% and 0.0%. These differences support reporting near-duplicate density when comparing retrieval results across corpora.

E A Stronger Retriever

§3's SEM retriever is `bge-base-en-v1.5`. Table 9 repeats the MathQA decomposition with `bge-large-en-v1.5` in its place, re-embedding the pool and the queries and re-ranking, with both conditions generated in one stage so the comparison uses conditions from the same generation stage.

Replacing bge-base with bge-large changes 74.3% of top-4 sets, with mean Jaccard similarity 0.571 between selections. The total gain decreases from +8.50 to +6.70 pp through changes in Δ_{pool} and Δ_{diff}, while Δ_{match} is +9.10 pp for both to two decimal places. This comparison is consistent with the near-duplicate account despite the differences in retrieved sets.

F The Decomposition, All Conditions

Table 10 applies Equation 1 to all eight corpus-retriever conditions, splitting each reported gain into its pool-reshaping, difficulty-coupling and query-matching parts.

G Observational Split and Dose-Response

MathQA queries whose nearest pool neighbour exceeds cosine 0.95 ($n = 646$) gain +18.11 pp from correct assignment; those below the threshold ($n = 354$) gain +2.54 pp. This approximately sevenfold difference holds within the same corpus, retriever, model and prompt format. The split is observational, and the subgroups differ in baseline

scale	permutation	Δ (pp)	95% CI (pp)	McNemar p	Holm p
SEM (embedding)					
outside	DER: query↔set deranged	+1.40	[-0.53, +3.34]	0.503	1.000
outside	DERSTRAT: deranged within difficulty stratum	+0.00	[-1.93, +1.93]	0.894	1.000
outside	ORDER: presentation order permuted	+0.76	[-1.14, +2.65]	0.480	0.480
inside: steps	STEPTOPO: steps reordered within the derivation DAG	-0.99	[-2.43, +0.38]	0.203	0.406
inside: steps	STEPVIOL: steps reordered against the derivation DAG	-2.43	[-4.17, -0.68]	0.008	0.023
inside: steps	STEPORDER: rationale step order permuted	-8.72	[-10.99, -6.52]	$8.9\cdot10^{-14}$	$3.6\cdot10^{-13}$
inside: numbers	NUMPERM: rationale numbers deranged	-18.57	[-21.23, -16.07]	$2.1\cdot10^{-43}$	$1.0\cdot10^{-42}$
inside: pairing	INTRA: problem↔rationale deranged in-set	-22.40	[-25.06, -19.90]	$5.5\cdot10^{-49}$	$1.6\cdot10^{-48}$
inside: pairing	INTRACORRECT: as INTRA: stated answer kept correct	-28.54	[-31.20, -25.93]	$1.6\cdot10^{-67}$	$6.4\cdot10^{-67}$
STRUCT (gold graph)					
outside	DER: query↔set deranged	-2.01	[-4.09, -0.04]	0.122	0.244
outside	DERSTRAT: deranged within difficulty stratum	-1.40	[-3.41, +0.53]	0.507	0.507
outside	ORDER: presentation order permuted	-0.30	[-2.20, +1.59]	0.817	0.817
inside: steps	STEPTOPO: steps reordered within the derivation DAG	-2.05	[-3.56, -0.53]	0.011	0.022
inside: steps	STEPVIOL: steps reordered against the derivation DAG	-5.38	[-7.13, -3.64]	$3.0\cdot10^{-9}$	$8.9\cdot10^{-9}$
inside: steps	STEPORDER: rationale step order permuted	-10.54	[-12.89, -8.19]	$2.7\cdot10^{-18}$	$1.1\cdot10^{-17}$
inside: numbers	NUMPERM: rationale numbers deranged	-17.89	[-20.39, -15.39]	$3.2\cdot10^{-42}$	$1.6\cdot10^{-41}$
inside: pairing	INTRA: problem↔rationale deranged in-set	-45.26	[-47.99, -42.61]	$1.4\cdot10^{-137}$	$4.3\cdot10^{-137}$
inside: pairing	INTRACORRECT: as INTRA: stated answer kept correct	-52.05	[-54.78, -49.39]	$1.2\cdot10^{-165}$	$5.0\cdot10^{-165}$

Table 4: The full permutation hierarchy on GSM8K, math-tuned model, both retrievers. Δ is against the unpermuted retrieved arm of the same retriever and stage. Intervals are paired bootstrap 95% over test items; McNemar is exact-binomial per replicate, reported at its least significant replicate; Holm is over the nine conditions of that retriever's hierarchy.

arm	demo tok	demo steps	$\cos(q,d)$	$\text{graph}(q,d)$
`rand0`	201.1	3.18	0.5754	0.4179
`struct_true`	199.2	3.24	0.5848	0.9410
`struct_derstrat0`	199.2	3.24	0.5744	0.5990
`struct_der0`	199.2	3.24	0.5742	0.4250
`sem_true`	206.5	3.45	0.7887	0.4472
`sem_intra0`	206.5	3.45	0.7887	0.4472

Table 5: Realised query–demonstration similarity, GSM8K. Structural match falls from the unpermuted arm through the stratified derangement to the unrestricted one, which sits at the uniform-random level. The INTRA arm is identical to SEM_TRUE on every column by construction, since it contains the same demonstrations. Full table in Appendix C.

difficulty, with random-demonstration accuracies of 59.9 and 55.1. The pruning experiment in §4 provides the intervention estimate.

Assignment gains across quartiles of realised query-demonstration similarity are +10.2, +8.2, +11.4 and +20.6 pp. The sequence is not monotone, although the top quartile has approximately twice the gain of the others. The fitted slope is +50.8 pp per unit cosine on MathQA and −5.7 on GSM8K. GSM8K's top quartile has mean similarity 0.851 and a gain of +1.36 pp, while MathQA's bottom quartile has mean similarity 0.828 and a gain of +10.2 pp. Mean similarity alone therefore does not explain the cross-corpus difference. At $\tau = 0.90$, nearest-neighbour similarities exceed the threshold for 2.4% of GSM8K queries and 79.8% of MathQA queries.

H Presentation-Order Sensitivity

The step-reordering conditions detect effects of about two percentage points. MathQA's order effect of −2.50 pp is larger but remains non-significant, limiting what can be inferred from that result. Lu et al. (2022) study classification tasks in which demonstration order changes the label sequence in the prompt. Our 4-shot mathematical chain-of-thought prompts have no corresponding label sequence, so the two studies evaluate different settings.

I Displacement Matching for the Derivation-DAG Conditions

A rationale's dependency graph specifies which results each step uses. Topology-preserving reorderings place each step after its dependencies; violating reorderings place at least one step before a required result. A linear chain permits no non-trivial topology-preserving reordering, limiting this

arm	demo tok	demo steps	$\cos(q, d)$	graph(q, d)
rand0	201.1	3.18	0.5754	0.4179
rand1	200.7	3.18	0.5738	0.4184
sem_der0	206.5	3.45	0.6039	0.4254
sem_der1	206.5	3.45	0.6024	0.4227
sem_derstrat0	206.5	3.45	0.6031	0.4254
sem_derstrat1	206.5	3.45	0.6036	0.4257
sem_intra0	206.5	3.45	0.7887	0.4472
sem_intra1	206.5	3.45	0.7887	0.4472
sem_intracorrect0	206.5	3.45	0.7887	0.4472
sem_intracorrect1	206.5	3.45	0.7887	0.4472
sem_true	206.5	3.45	0.7887	0.4472
struct_der0	199.2	3.24	0.5742	0.4250
struct_der1	199.2	3.24	0.5737	0.4247
struct_derstrat0	199.2	3.24	0.5744	0.5990
struct_derstrat1	199.2	3.24	0.5749	0.6050
struct_intra0	199.2	3.24	0.5848	0.9410
struct_intra1	199.2	3.24	0.5848	0.9410
struct_intracorrect0	199.2	3.24	0.5848	0.9410
struct_intracorrect1	199.2	3.24	0.5848	0.9410
struct_true	199.2	3.24	0.5848	0.9410

Table 6: Nuisance matching and realised similarity, every arm, GSM8K. Columns are means over test items: prompt tokens per demonstration, gold steps per demonstration, query–demonstration embedding cosine, and query–demonstration graph similarity. Higher is more similar; both similarities lie in $[0, 1]$.

corpus	embedder	.85	.90	.92	.95	.97	.99
GSM8K	bge-base	21.8	2.4	0.8	0.0	0.0	0.0
	bge-small	13.7	1.1	0.3	0.0	0.0	0.0
	MiniLM-L6	1.0	0.0	0.0	0.0	0.0	0.0
MathQA	bge-base	91.0	79.8	73.9	64.6	52.6	26.4
	bge-small	92.6	81.6	75.9	64.6	51.8	23.8
	MiniLM-L6	69.0	59.9	55.5	44.5	32.7	12.1

Table 7: Near-duplicate density, the share of test queries whose nearest pool neighbour clears the cosine threshold, under three embedders. bge-base is the retriever of §3; MiniLM-L6 is a different family with different training and 384 dimensions. GSM8K holds no near-duplicate under any embedder at any threshold from 0.95 up.

corpus	pool	test	.85	.90	.95	.99
SVAMP	700	300	85.3	73.7	64.3	54.7
MathQA	8,000	1,000	69.0	59.9	44.5	12.1
ASDiv	2,075	230	28.7	14.3	5.7	4.3
AQuA	12,000	254	9.1	3.5	3.1	2.4
MAWPS	1,089	520	1.0	0.0	0.0	0.0
GSM8K	7,473	1,319	1.0	0.0	0.0	0.0

Table 8: Near-duplicate density across math word-problem corpora using `all-MiniLM-L6-v2`. Entries give the percentage of test queries whose nearest training-pool neighbour exceeds the column's cosine threshold. GSM8K and MathQA repeat the MiniLM-L6 measurements in Table 7.

comparison to 38.8% of pool rationales and 42.9% of the demonstrations shown. Each rationale is permuted in both conditions or neither, leaving the prompts byte-identical wherever the comparison is unavailable. Violating the dependency structure reduces accuracy by an additional 1.4 pp. Mean step displacement is 1.59 positions in that condition and 0.81 in the topology-preserving condition. Sensitivity to displacement could therefore explain the ordering without sensitivity to graph structure. A displacement-matched comparison is required to separate these effects.

J Retrieved-Pool Concentration

Table 11 shows concentration in the retrieved demonstration distributions, with no significant Δ_{pool} effect in the four conditions considered here. Graph matching is more concentrated because the pool contains 2100 distinct gold operator signatures. Queries with the same signature can therefore receive the same four examples from a small equivalence class. Fixed random tie-breaking changes the median selected pool index from 717 to 3746 while leaving the Gini coefficient at 0.866. Embedding retrieval is less concentrated but raises mean demonstration step count from 3.18 to 3.45 and lowers the share of short ($\leq$ 2-step) demonstrations from 0.354 to 0.274.

retriever	Δ_{total}	Δ_{pool}	Δ_{diff}	Δ_{match}
bge-base-en-v1.5	$+8.50$	$+0.40$	-1.00	$+9.10$
bge-large-en-v1.5	$+6.70$	-0.70	-1.70	$+9.10$

Table 9: MathQA decomposition for two retrievers using Qwen2.5-Math-1.5B, with both evaluated in one generation stage. The selected top-4 sets differ for 74.3% of queries, with mean Jaccard similarity 0.571. Both matching terms equal $+9.10$ pp to two decimal places and are significant at $p < 10^{-4}$. All effects are in percentage points; $n = 1000$.

K Output Characteristics Under Permutation

Under INTRA, the model produces a well-formed numeric answer in 100.0% of cases, with mean generation length 86.0 tokens compared with 89.2 in the unpermuted condition. The fraction of predictions equal to a printed demonstration answer rises from 6.4% to 17.5% for embedding retrieval and to 43.3% for structural retrieval. Structural retrieval selects examples with similar derivations, which may make their answers more likely to be copied. Table 12 reports accuracy, well-formed answer rate, copy rate and generation length for each condition.

corpus	retr.	Δ_{total}	Δ_{pool}	Δ_{diff}	Δ_{match}
GSM8K, general 1.5B	sem	-0.30	+0.30	-0.23	-0.38
GSM8K, general 1.5B	struct	+0.91	+0.61	+0.23	+0.08
GSM8K, math 1.5B	sem	-0.53	+0.87	-1.40	+0.00
GSM8K, math 1.5B	struct	+2.12	+0.11	+0.61	+1.40
GSM8K, math 7B	sem	-0.08	-1.10	+0.80	+0.23
GSM8K, math 7B	struct	+1.14	+0.76	+0.57	-0.19
MathQA, math 1.5B	sem	+10.20	-1.65	-0.75	+12.60
MathQA, math 1.5B	struct	+4.10	+0.50	-1.80	+5.40

Table 10: The decomposition of Equation 1. The identity is exact; the three terms are rounded independently, so a printed row can differ from its printed Δ_{total} by 0.01. All four terms are in percentage points.

population	unique	Gini	mean steps	≤ 2 steps (share)	top 1% share
whole pool	7473	–	3.17	0.358	–
uniform draw	3733	0.609	3.18	0.354	–
SEM	2941	0.729	3.45	0.274	9.7%
STRUCT	1968	0.866	3.24	0.334	26.1%

Table 11: Demonstration selection statistics on GSM8K, computed from retrieval indices without model generation.

arm	acc (%)	well-formed	copies a demo	gen. tok
rand0	73.46	100.0%	3.7%	90.3
rand1	75.44	100.0%	3.9%	91.7
sem_der0	75.89	100.0%	3.3%	90.2
sem_der1	74.75	100.0%	3.6%	91.0
sem_derstrat0	74.15	100.0%	4.8%	89.9
sem_derstrat1	73.69	100.0%	3.1%	89.6
sem_intra0	50.95	100.0%	18.2%	86.0
sem_intra1	52.08	100.0%	16.8%	84.9
sem_intracorrect0	46.10	100.0%	21.7%	86.5
sem_intracorrect1	44.66	100.0%	22.6%	85.4
sem_true	73.92	100.0%	6.4%	89.2
struct_der0	74.45	100.0%	3.7%	91.5
struct_der1	74.68	99.9%	3.1%	92.0
struct_derstrat0	74.60	100.0%	3.0%	94.4
struct_derstrat1	75.74	100.0%	3.0%	92.5
struct_intra0	31.99	100.0%	42.9%	90.5
struct_intra1	30.63	100.0%	43.7%	89.0
struct_intracorrect0	25.09	100.0%	50.3%	90.1
struct_intracorrect1	23.96	99.9%	52.0%	90.8
struct_true	76.57	100.0%	5.9%	91.3

Table 12: Failure mode per arm on GSM8K. 'Copies a demo' is the fraction of predictions equal to one of the answers printed in the demonstration block.

InvariantSpec: Benchmarking LLM Autoformalization of Conservation and Dissipation Laws into Verifiable State-Space Constraints

Hikaru Matsuoka
University of Nagasaki
RIKEN AIP
matsuokah@sun.ac.jp

Abstract

Mathematical descriptions of dynamical systems denote *conjunctions* of proof obligations—the advection–diffusion equation conserves mass *and* dissipates the L^2 norm—yet common task designs ask language models to summarize a system with a single structure label, a target that is wrong by construction for mixed conservation–dissipation systems. We recast the problem as *autoformalization*: translating natural language, equations, and stated assumptions into a typed formal specification whose atoms are machine-checkable matrix conditions on the generator of a linear state-space model ($A^\top c = 0$, $A^\top M + MA = 0$, $A^\top M + MA \preceq 0$, $A^\top J + JA = 0$). We introduce **InvariantSpec**, a benchmark of 12 dynamical systems $\times$ 5 linguistic variants $\times$ 3 modality conditions (180 inputs) with verifier-executable gold specifications, contrastive minimal pairs (boundary conditions, conserved vs. non-increasing, sentence order, quadratic vs. cubic invariants), and two systems whose correct answer is *not representable by a fixed linear generator*. A symbolic verifier checks every predicted obligation and returns typed counterexample feedback; verified specifications are then *compiled* into constrained generators, including a new parameterization that simultaneously conserves a linear quantity and dissipates a quadratic one. Across four frontier LLMs, single-label structure selection has exact specification match 0.00—a ceiling imposed by the answer space, not the models—while autoformalization with one round of verifier-guided refinement reaches up to 0.88 with soundness 0.98–1.00; the residual gap is dominated by unstated side conditions (assumption extraction ranges 0.23–0.93), not by obligation algebra. Downstream, one-step MSE does not predict long-rollout structural fidelity: spec-compiled generators hold every verified invariant to machine precision over 1000-step rollouts, while unconstrained models with equal or better MSE violate them by up to 10^{17}.

1 Introduction

Consider the periodic advection–diffusion equation $u_t + c\,u_x = \kappa\,u_{xx}$. Its physics is a conjunction of two facts of opposite algebraic type: the total mass $\int u\,dx$ is *exactly conserved* (an equality), while the L^2 norm $\int u^2\,dx$ is *dissipated* (an inequality). A task that asks a model to choose *one* structure class for this system—"unitary" (norm-preserving) or "contractive" (norm-decaying)—has no correct answer: whichever label the model picks discards half of the mathematics. In pilot studies with such a design we repeatedly saw models identify *both* properties in their rationale (*"the system requires a combination of conservation (mass) and dissipation (L^2 norm)…"*) and then commit to one label. This is not only a model failure; it is a *task-design failure*: collapsing a specification into one label makes the gold standard itself unsound.

This paper treats the problem as *autoformalization* (Wu et al., 2022): translating mathematical natural language, equations, and stated assumptions into a typed formal specification that a machine can verify. Our target formalism is deliberately lightweight. For a linear state-space model (SSM) $\dot{h} = Ah$ with discrete map $\Phi_\Delta = \exp(\Delta A)$—the computational core of the S4/S5/Mamba family (Gu et al., 2022; Smith et al., 2023; Gu and Dao, 2024; Dao and Gu, 2024)—every conservation or dissipation statement reduces to a *matrix condition on the generator*: conservation of a linear quantity $c^\top h$ is $A^\top c = 0$; conservation of a quadratic quantity $\frac{1}{2}h^\top Mh$ is the Lyapunov equality $A^\top M + MA = 0$; dissipation is the semidefinite inequality $A^\top M + MA \preceq 0$; preservation of a symplectic two-form J is $A^\top J + JA = 0$. These conditions are decidable by elementary linear algebra, so the verifier returns not just *false* but a *counterexample witness*: a concrete state at which the claimed invariant measurably changes—the verifier-grounded contract of formal mathemat-

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 101–111
October 29, 2026 ©2026 Association for Computational Linguistics

ics (Jiang et al., 2023; Azerbayev et al., 2023) at the applied end of mathematical language (Meadows and Freitas, 2023; Huang et al., 2025).

The formal target also sharpens a distinction natural language habitually blurs: *symplecticity and energy conservation are different obligations*. The plausible claim "$A = JS$ with S symmetric preserves $\frac{1}{2}h^\top Mh$ whenever $J^{-1}M$ commutes with S" is false: for canonical J, $S = I$, $M = \mathrm{diag}(1, 2)$ the premise holds trivially, yet the verifier exhibits a state with $\frac{d}{dt}\frac{1}{2}h^\top Mh = -\frac{1}{2} \neq 0$ (Proposition 1). Informal plausibility is exactly what a proof-obligation checker is for.

Contributions. **(1)** A typed specification language whose atoms are machine-checkable proof obligations for conservation, dissipation, two-form preservation, required assumptions, and *representability*—whether a fixed linear generator can realize the dynamics at all (§2). **(2)** INVARIANTSPEC: 12 systems × 5 linguistic variants × 3 modalities with verifier-executable gold specifications and contrastive minimal pairs (boundary conditions, polarity, sentence order, representability) (§3). **(3)** A symbolic verifier with counterexample search and a typed error taxonomy, used for evaluation and for one round of verifier-guided refinement, plus a *constraint compiler* from verified specifications to constrained generators—including one that provably conserves a linear quantity while dissipating a quadratic one, replacing the unsound "unitary or contractive" choice with sound constraint composition (§4). **(4)** An evaluation of four frontier LLMs with soundness/completeness metrics and cluster-bootstrap intervals, and a downstream SSM case study showing that one-step MSE does not predict long-rollout structural fidelity, whereas compiled constraints hold invariants at machine precision (§5–§6).

2 Typed Formal Specifications

Setting. We work with autonomous linear dynamics $\dot{h} = Ah$ on $\mathbb{R}^n$ (or $\mathbb{C}^n$), with exact discrete flow $\Phi_\Delta = \exp(\Delta A)$. A *specification* is a conjunction of atomic *proof obligations*, each tying a named physical quantity to a matrix condition on A (Table 1), together with a list of *assumptions* (side conditions such as boundary conditions) and a *representability* judgment.

All obligations are grounded by one calculation.

Obligation	**Condition on A**
`preserve_linear`(c)	$A^\top c = 0$
`preserve_quadratic`(M)	$A^\top M + MA = 0$
`dissipate_quadratic`(M)	$A^\top M + MA \preceq 0$
`preserve_two_form`(J)	$A^\top J + JA = 0$
`preserve_complex_norm`	$A^* + A = 0$
`requires`(a)	side condition a
`unsupported`(r)	no fixed linear A exists

Table 1: The specification language. Quantities are $c^\top h$, $\frac{1}{2}h^\top Mh$ ($M{=}M^\top$), a nondegenerate skew form J, and the complex norm. Arguments are *canonical form names* (`mass`, `l2_norm`, `energy`, ...) resolved by the verifier against the system's discretization: grading is by the mathematical object, not string identity.

Theorem 1 (Quadratic quantities under linear flow)**.** *Let $M = M^\top$. Then for all t,*

$$\frac{d}{dt}\left(e^{tA^\top}Me^{tA}\right) = e^{tA^\top}\left(A^\top M + MA\right)e^{tA}.$$

Consequently (i) $A^\top M + MA = 0 \iff \Phi_\Delta^\top M\Phi_\Delta = M$ for all Δ; (ii) $A^\top M + MA \preceq 0 \implies \Phi_\Delta^\top M\Phi_\Delta \preceq M$ for all $\Delta \geq 0$; and (iii) $A^\top c = 0 \iff \Phi_\Delta^\top c = c$ for all Δ.

Proof. The identity is the product rule. For (ii), fix v and note $t \mapsto v^\top e^{tA^\top}Me^{tA}v$ has nonpositive derivative pointwise, hence is nonincreasing; (i) and (iii) follow by integrating and by $\frac{d}{dt}e^{tA^\top}c = e^{tA^\top}A^\top c$. □

Theorem 1 is what makes the DSL *verifiable*: each continuous-time obligation transports to an exact certificate for the discrete map used by the trained model, with no scheme-dependent error term. Conservation and dissipation live in one framework but with different logical strength (equality vs. semidefinite inequality), which the evaluation keeps separate throughout.

Symplecticity is not energy conservation. Natural language freely slides between "symplectic", "Hamiltonian", and "energy-conserving". The formal picture is sharper.

Proposition 1. *Let $J^\top = -J$, $J^2 = -I$, $S = S^\top$, and $A = JS$. Then $A^\top J + JA = 0$ always (the two-form is preserved), and for symmetric M,*

$$A^\top M + MA = 0 \iff MJS = SJM.$$

In particular $M = S$ is always conserved, but a generic M is not: for canonical J in $\mathbb{R}^2$, $S = I$, $M = \mathrm{diag}(1, 2)$ we get $A^\top M + MA = \left(\begin{smallmatrix} 0 & -1 \\ -1 & 0 \end{smallmatrix}\right) \neq 0$, and at the witness state

$v = \frac{1}{\sqrt{2}}(1,1)^\top$ *the "invariant" changes at rate* $\frac{d}{dt}\frac{1}{2}v^\top Mv = -\frac{1}{2}$.

Proof. $A^\top J + JA = -SJJ + JJS = S - S = 0$. For the equivalence, $A^\top M + MA = -SJM + MJS$, which vanishes iff $MJS = SJM$. The instance is a direct computation. □

The correct condition couples all three of M, J, S; the plausible weaker premise "$J^{-1}M$ commutes with S" holds trivially above ($S = I$) while the conclusion fails. The verifier refutes such claims constructively with the witness v—the message the refinement loop (§4) feeds back. `preserve_two_form` and `preserve_quadratic` are therefore separate atoms; a Hamiltonian system lists both.

Representability. A specification finally judges whether the dynamics is realizable by a fixed linear generator at all: `exact_linear`, `approximate_linear`, or `not_representable`. Euler's rigid-body equations ($\dot{\Pi} = \Pi \times \Omega$: state-dependent generator) and KdV (quadratic advection, cubic Hamiltonian) are `not_representable`: the honest answer is to *refuse the surrogate class*, removing the artifact of single-label designs in which nonlinear systems must be force-mapped onto some class (KdV → "unitary") to be gradable at all. Models emit specifications as schema-validated JSON; Appendix A shows the advection–diffusion gold specification.

3 The InvariantSpec Benchmark

Systems and gold specifications. Table 2 lists the 12 base systems: Hamiltonian, dissipative, and—crucially—*mixed* systems whose specification is a nontrivial conjunction, plus two nonlinear systems whose gold answer is `not_representable`. PDEs are discretized on periodic or Dirichlet grids (central differences; $n \leq 32$); each system exposes its named forms (mass vector, energy matrix, graph Laplacian, canonical J), and obligations are graded *semantically*: a prediction matches gold iff both resolve to numerically proportional matrices under the same condition type. An oracle answering with gold specifications scores 1.000 on every metric—the noise floor is zero.

Linguistic variants and modalities. Each system is described in five registers (textbook, applied/engineering, terse lecture-note, verbose narrative, and a statement-style variant used by the minimal pairs), each under three modality conditions—*equation-only*, *text-only*, and *full* (equation + text + explicit assumptions)—yielding $12 \times 5 \times 3 = 180$ inputs. Every text variant uniquely determines the system, so gold is recoverable in every condition; what varies is whether structure must be derived from symbols, extracted from prose, or reconciled across both.

Minimal pairs. Four contrastive pairs isolate one linguistic dimension each, holding everything else fixed:

Boundary condition. The same heat equation with "periodic" vs. "homogeneous Dirichlet" boundary conditions: mass conservation must appear in the first specification and be absent from the second.

Polarity. "the L^2 norm … is conserved as well" vs. "… is non-increasing" in otherwise matched sentences (advection vs. advection–diffusion): one content word flips an equality obligation to an inequality.

Sentence order. "Mass is conserved, while the L^2 energy dissipates" vs. "The L^2 energy dissipates, although the mass remains constant"—the same system, so the outputs must be formally equivalent (probing the primacy effect seen in single-label pilots).

Representability. Matched sentences "conserves a *quadratic* energy … *linear* vector field" vs. "conserves a *cubic* energy … *nonlinear* vector field" (anisotropic oscillator vs. KdV): the invariant's degree decides `exact_linear` vs. `not_representable`.

A pair is scored *consistent* only if the contrast is resolved correctly on both sides.

4 Autoformalization with a Verifier in the Loop

4.1 Three task designs

Baseline A: single structure label. The model picks one of {*symplectic*, *unitary*, *contractive*, *unconstrained*, *vanilla*}. To grade this baseline in the same currency as the others, each label is mapped to the specification it implies (unitary → `preserve_quadratic`(L^2); contractive → `dissipate_quadratic`(L^2); symplectic → `preserve_two_form`(J); unconstrained/vanilla → no obligations). This makes the design's ceiling

System	Gold obligations	Assumptions	Repr.
harmonic oscillator	`pres_quad`(energy), `pres_2form`(J)	—	exact
anisotropic oscillator	`pres_quad`(energy, non-Euclidean), `pres_2form`(J)	—	exact
damped oscillator	`diss_quad`(energy)	—	exact
periodic advection	`pres_lin`(mass), `pres_quad`(L^2)	periodic BC	exact
periodic heat	`pres_lin`(mass), `diss_quad`(L^2)	periodic BC	exact
Dirichlet heat	`diss_quad`(L^2); mass *not* conserved	Dirichlet BC	exact
periodic advection–diffusion	`pres_lin`(mass), `diss_quad`(L^2)	periodic BC	exact
wave equation	`pres_quad`(energy), `pres_lin`(momentum), `pres_2form`(J)	periodic BC	exact
linear Schrödinger	`pres_complex_norm`	periodic BC	exact
graph consensus	`pres_lin`(mean), `diss_quad`(disagreement)	connected graph	exact
free rigid body	—	—	**not repr.**
KdV	—	—	**not repr.**

Table 2: The 12 base systems and their gold specifications. Six involve more than one simultaneous obligation; two require refusing the linear surrogate class altogether. Every gold specification is validated by the symbolic verifier against the system's discretized generator (all obligations pass; all discrete certificates of Theorem 1 hold to $< 10^{-12}$).

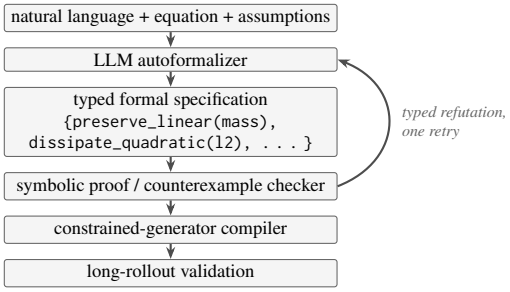


Figure 1: Pipeline. The LLM produces a typed specification; the verifier checks every obligation against the system's discretized generator and either certifies it or returns a typed counterexample message for one refinement round; verified specifications compile into constrained generators whose long rollouts are validated numerically.

explicit: no label maps to a conjunction, so exact match on any multi-obligation system is impossible *regardless of model quality*.

Baseline B: direct autoformalization. The model emits the JSON specification directly. We use schema-constrained decoding (`response_format=json_schema`) whenever the provider supports it, so that interface failures are separated from semantic failures by construction rather than by parser forensics.

Proposed: autoformalization + verifier-guided refinement. The specification from Baseline B is checked by the verifier. On failure, the model receives a typed report and revises *once*. Example report on a polarity error: *"[polarity]* $A^\top M + MA = 0$ *fails but* $A^\top M + MA \preceq 0$ *holds: the quadratic quantity 'L^2 norm' is dissipated, not conserved (at the witness state,* $\frac{d}{dt}Q = -3.1 \times 10^{-1}$*)—change* `preserve_quadratic` *to* `dissipate_quadratic`*."*

4.2 The verifier

Given the system's discretized generator (mechanically derived from the equation—no gold labels consulted), the verifier resolves each named form, checks the matrix condition (equalities by residual norm, inequalities by extremal eigenvalue), and rechecks the discrete certificates of Theorem 1 on Φ_Δ; small systems are re-checked in exact rational arithmetic (SymPy). Every failed check yields (i) a *witness state* at which the claimed invariant demonstrably changes and (ii) an error label from the taxonomy of §7: *quantity identification*, *polarity*, *algebraic typing*, *missing assumption*, *conjunction collapse*, or *representability*. The assumption-completeness check is *evaluation-only*: never included in refinement feedback, so the loop receives no annotation signal beyond the executable model of the system itself.

4.3 Metrics

Let $\hat{O}$ be the predicted and O^* the gold obligation set, matched semantically (§3). We report *soundness* $= |\{o \in \hat{O} : \text{verifier-true}\}|/|\hat{O}|$ and *completeness* $= |\{o \in O^* : \exists\, \hat{o} \text{ matching}\}|/|O^*|$, plus *exact specification match* (mutual matching, correct representability, all required assumptions stated), obligation-level macro-F_1, assumption extraction accuracy, unsupported-case accuracy, minimal-pair consistency, verifier pass rate, and one-retry correction rate. Soundness is graded by the *verifier*, not by gold matching, so a true-but-unlisted obligation (e.g. L^2 conservation for the Schrödinger flow) is

never penalized as false. Paraphrases and modalities of one system are never treated as independent samples: we aggregate per base system and report 95% CIs from a cluster bootstrap over the 12 systems. Parse failures are tracked separately from all semantic metrics.

4.4 Compiling verified specifications

A verified specification is a conjunction of linear *equality* constraints and PSD *inequality* constraints on A, which we compile into a generator parameterization satisfying every obligation *for all parameter values*—structure is parameterized, not learned.

Proposition 2 (Constraint compilation)**.** *Let $K^\top = -K$ and let R be arbitrary.* ***(P1)*** *If $M \succ 0$ and $A = M^{-1}K$ then $A^\top M + MA = 0$.* ***(P2)*** *If $M \succ 0$ and $A = M^{-1}(K - R^\top R)$ then $A^\top M + MA = -2R^\top R \preceq 0$.* ***(P3)*** *Let $P = I - \frac{cc^\top}{c^\top c}$ and $A = PKP - PR^\top RP$. Then $A^\top c = 0$ and $A^\top + A = -2PR^\top RP \preceq 0$: the linear quantity $c^\top h$ is conserved while $\|h\|^2$ is dissipated, simultaneously.* ***(P3′)*** *More generally, for symmetric $G \succeq 0$, $A = (PKP - PR^\top RP)\,G$ conserves $c^\top h$ and dissipates $h^\top Gh$.*

Proof. (P1) $A^\top M + MA = K^\top M^{-1}M + MM^{-1}K = K^\top + K = 0$. (P2) identical, leaving $-2R^\top R$. (P3) $Pc = 0$ gives $A^\top c = 0$; the symmetric part of A is $-PR^\top RP$. (P3′) $A^\top c = G(\cdots)^\top c = 0$ and $A^\top G + GA = G(B + B^\top)G = -2(RPG)^\top(RPG)$ with $B = PKP - PR^\top RP$. □

P3 directly serves the mixed systems (heat, advection–diffusion, consensus) that single-label designs cannot express: mass is conserved *and* the L^2 norm is non-increasing, by construction. Pure-equality conjunctions (e.g. two-form *and* non-Euclidean energy) are compiled generically: each equality obligation is linear in A, so a basis of the joint nullspace is computed once by SVD and A is parameterized inside it; property-based tests confirm every compiled class satisfies its obligations for random parameters and contains the true generator. Specifications judged `not_representable` refuse compilation—the honest outcome for KdV and the rigid body.

5 LLM Experiments

Models and decoding. We evaluate four frontier models through a single API gateway: GPT-5.6-terra, Claude Sonnet 5, Gemini 3.6 Flash, and DeepSeek V3.2 (temperature 0, low reasoning effort, schema-constrained JSON decoding). Each model answers all 180 inputs under all three task designs of §4.1; the refinement round is triggered only when the verifier rejects the first specification. Interface failures that survive schema-constrained decoding (provider errors, truncation) are reported as parse failures and excluded from semantic metrics; they are rare ($\leq 2.8\%$, zero for most cells).

Single-label designs have a semantic ceiling. Table 3 quantifies the task-design failure that motivated this work. Graded through the specifications they imply, the single-label baselines reach soundness 0.73–0.76 but completeness only 0.39–0.40: a label can assert one true obligation while necessarily dropping the rest of the conjunction, and exact specification match is 0.000 for *every* model—not because the models are weak, but because the answer space cannot express the gold standard. The label space also cannot express *refusal*: unsupported-case accuracy is 0.00 across the board, since every available label asserts some structure for systems that have no linear realization.

Direct autoformalization recovers the conjunction. Under the typed DSL, exact specification match rises to 0.55–0.85 (best: Gemini 3.6 Flash), soundness to 0.95–0.98, and completeness to 0.94–0.98. The verifier already certifies 91–96% of first-attempt specifications, and *every* model attains perfect unsupported-case accuracy: given a vocabulary for refusing, frontier models reliably recognize that Euler's equations and KdV admit no fixed linear generator—the same systems that single-label designs force onto a least-wrong class.

Verifier refinement makes assertions true. One round of typed counterexample feedback drives soundness to 0.98–1.00: after refinement, essentially every obligation asserted is *true*, and the *Fix* column shows that 70–100% of the few first-attempt verifier failures (7–10 per model out of 180) are fully repaired by a single retry. Exact match improves for three models (best 0.88, Gemini 3.6 Flash). The exception is instructive: GPT-5.6-terra's exact match *decreases* slightly under refinement because rewriting occasionally drops assumption fields—which our no-leak design deliberately excludes from verifier feedback (§4.2)—trading a verifier-visible improvement for a verifier-invisible regression.

Model	Method	PF%	Exact match [95% CI]	Sound.	Compl.	mF_1	Asm.	Unsup.	VPass	Fix
GPT-5.6-terra	single label	0.0	0.000 [0.00,0.00]	0.761	0.403	0.653	0.00	0.00	0.76	–
	direct	0.0	0.550 [0.35,0.76]	0.975	0.981	0.937	0.26	1.00	0.95	–
	+verifier	0.0	0.533 [0.34,0.75]	1.000	0.991	0.957	0.23	1.00	1.00	1.00
Claude Sonnet 5	single label	0.0	0.000 [0.00,0.00]	0.750	0.403	0.653	0.00	0.00	0.75	–
	direct	0.0	0.744 [0.58,0.89]	0.978	0.979	0.954	0.64	1.00	0.96	–
	+verifier	0.0	0.767 [0.63,0.90]	0.997	0.978	0.968	0.69	1.00	0.99	0.86
Gemini 3.6 Flash	single label	0.0	0.000 [0.00,0.00]	0.728	0.394	0.641	0.00	0.00	0.73	–
	direct	0.0	0.850 [0.73,0.96]	0.968	0.975	0.948	0.93	1.00	0.94	–
	+verifier	0.0	0.883 [0.78,0.97]	1.000	0.989	0.970	0.93	1.00	1.00	1.00
DeepSeek V3.2	single label	0.0	0.000 [0.00,0.00]	0.761	0.389	0.641	0.00	0.00	0.76	–
	direct	0.6	0.763 [0.63,0.89]	0.947	0.938	0.903	0.84	1.00	0.91	–
	+verifier	2.8	0.796 [0.69,0.90]	0.980	0.936	0.930	0.84	1.00	0.98	0.70

Table 3: Main results (180 inputs per cell; means of per-system means; cluster-bootstrap CIs over the 12 systems). PF = parse/interface failures (excluded from semantic metrics); mF_1 = obligation macro-F_1; Asm. = assumption extraction; Unsup. = accuracy on the two non-representable systems; VPass = verifier pass rate; Fix = failed first attempts corrected by one verifier round. Single-label rows are graded through the specification each label implies; their exact match is 0 *by construction* on multi-obligation systems.

	Exact match by modality			Minimal pairs (consistent?)			
Model	eq. only	text only	full	BC	polarity	order	repr.
GPT-5.6-terra	0.40	0.47	0.73	✓	✓	✓	✓
Claude Sonnet 5	0.75	0.63	0.92	✓	✓	✓	✓
Gemini 3.6 Flash	0.83	0.87	0.95	✓	✓	✓	✓
DeepSeek V3.2	0.71	0.73	0.95	✓	✓	✓	×

Table 4: Modality ablation and minimal-pair consistency for the verifier-refined method. A pair counts as consistent only if the contrast is resolved correctly on both sides (§3).

Assumptions, not algebra, separate the models. The *Asm.* column exposes the largest capability gap: assumption extraction ranges from 0.93 (Gemini) down to 0.23 (GPT-5.6-terra), while obligation-level metrics are tightly clustered. For GPT-5.6-terra, 61 of 84 non-exact refined items have *perfect obligations* and fail only by omitting the boundary-condition side conditions their own specification depends on. Formalizing the mathematics is largely solved for these systems; formalizing its *context of validity* is not.

Modalities and minimal pairs. Table 4 separates reading mathematics from reading about mathematics. The full condition is best for every model (0.73–0.95 exact); equation-only is generally hardest (GPT-5.6-terra drops to 0.40), since invariants must be derived rather than extracted—e.g. noticing that the Dirichlet Laplacian does not annihilate the ones vector. With typed specifications, the minimal pairs are resolved almost uniformly: all models pass the boundary-condition, polarity, and sentence-order contrasts, and all pass representability under direct generation (DeepSeek regresses on it after refinement). Under the single-label design the boundary-condition and representability contrasts are *structurally unanswerable* and every model fails them: the contrastive failures the old design exhibited were largely artifacts of its answer space. The pairs still matter—they pinpoint the one-token distinctions (*conserved* vs. *non-increasing*; *periodic* vs. *Dirichlet*) that decide between equality and inequality constraints downstream.

6 Downstream Case Study: Compiled Constraints in Trained SSMs

The specification is the contract a downstream surrogate must honor. We close the loop by training linear SSMs on the ten linear benchmark systems and measuring whether declared structure survives training and long rollouts. For each system, teacher trajectories come from the exact flow (32 initial conditions, 80 steps, $\Delta = 0.05$); three parameterizations train on one-step MSE (Adam, 200 epochs, float64): *unconstrained* A; *vanilla* $A = -\operatorname{diag}(\exp\theta)$, the standard stability-first SSM rule; and *spec*, the gold specification compiled by Proposition 2. Rollouts run 1000 steps from 8 held-out initial conditions.

Obligation-appropriate error metrics. Conservation and dissipation must not share a "drift" number: drift is meaningless for a quantity that is *sup-*

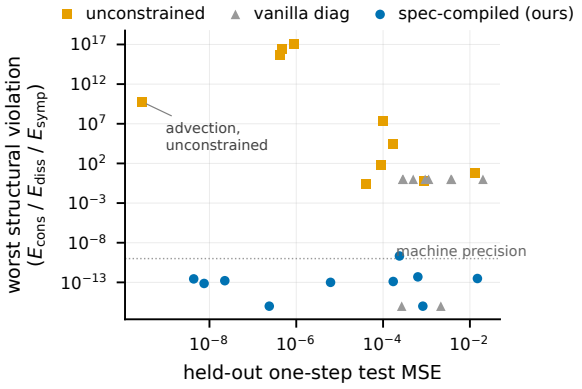


Figure 2: One-step test MSE does not predict long-rollout structural fidelity. Each point is one (system, parameterization); y is the worst violated obligation metric over a 1000-step rollout. Unconstrained models occupy the upper region at every MSE level (up to 1.2×10^{17}); spec-compiled models sit at machine precision across the entire MSE range.

posed to decrease. For a conserved quantity I, a dissipated quantity Q, and a two-form J we report

$$E_{\text{cons}} = \max_{t \le T} \frac{|I(h_t) - I(h_0)|}{|I(h_0)| + \varepsilon},$$

$$E_{\text{diss}} = \frac{\sum_{t<T} \left[Q(h_{t+1}) - Q(h_t)\right]_+}{Q(h_0) + \varepsilon},$$

$$E_{\text{symp}} = \frac{\|\Phi^\top J \Phi - J\|_F}{\|J\|_F},$$

i.e. worst-case relative drift for equalities, accumulated *increase* for inequalities (decrease is physics, not error), and the discrete two-form defect of the learned map itself.

Results. Table 5 and Figures 2–3 give three findings. **(i) One-step MSE does not predict structure:** on periodic advection the unconstrained model attains the study's best test MSE (2.8×10^{-10}) yet drifts mass by $\sim 10^5$ and L^2 by $\sim 10^9$ over 1000 steps; on periodic heat it reaches 4.6×10^{-7} MSE while *increasing* the supposedly dissipated L^2 norm by 2.5×10^{16}. **(ii) Compiled conjunctions cost little and can help:** spec-compiled models hold every equality obligation to $\le 2.1 \times 10^{-10}$ and never violate an inequality ($E_{\text{diss}} = 0$), with test MSE within a small factor of unconstrained—on diffusive systems *better* (heat: 4.3×10^{-9} vs. 4.6×10^{-7}), because the constraint class excludes spurious expanding directions that one-step fitting cannot rule out. **(iii) Mixed conservation–dissipation is realized exactly:** Figure 3 shows both obligations of advection–diffusion holding simultaneously—the regime the unitary/contractive dichotomy cannot express.

7 Error Analysis

Because every failed check carries a typed verifier label, error analysis is a count, not a reading exercise. Across all four models' *direct* outputs (719 parseable specifications), the verifier and gold comparison yield 179 error events, with a distribution that inverts the usual expectations:

Missing assumption (119). By far the dominant class: obligations correct but the boundary-condition side condition unstated (67 events for GPT-5.6-terra alone, 6 for Gemini). Invisible to a free-form grader; caught by the typed field.

Algebraic typing (45). Quantities that do not type-check in the DSL. The recurrent pattern (18 events on Dirichlet heat) is physically astute: models want to assert that the *mass decays*, but a linear functional's decay admits no semidefinite certificate, so the DSL rejects it. These errors mark a real expressiveness boundary of the formalism (see Limitations), not model confusion; the verifier's message explains the boundary and suggests the certifiable alternative.

Conjunction collapse (10). Exactly one obligation of a multi-obligation gold specification emitted—the single-label habit persisting into the richer format, now rare rather than forced.

Quantity identification (3) and polarity (1). Nearly extinct: models almost never claim a false invariant or confuse conserved with dissipated *when the answer space lets them assert both*; the polarity failures seen in pilot studies were overwhelmingly collapse artifacts.

Representability (1). One hallucination in the full run—a model declaring a frictionless mass–spring system `not_representable` because the spring "could be nonlinear"; the two-sided check (refusing linear systems is also an error) remains a necessary guard.

In aggregate: frontier models formalize the *algebra* of these systems nearly perfectly once the task is posed as autoformalization; residual failures concentrate in *pragmatics* (the context of validity) and in the *formalism's expressiveness limits*—exactly

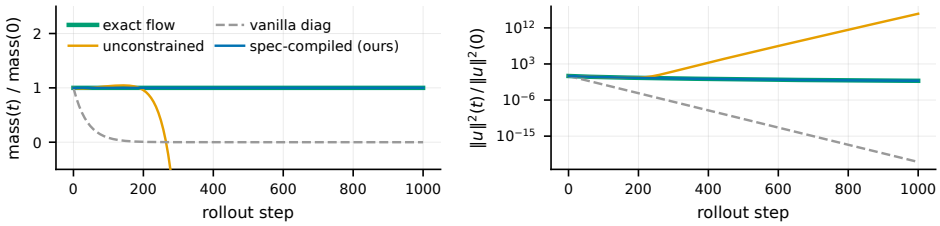


Figure 3: The two obligations of periodic advection–diffusion along a rollout. **Left:** mass ratio (equality); **right:** L^2 ratio, log scale (inequality). The spec-compiled generator (P3) tracks the exact flow; the unconstrained model loses mass catastrophically and grows L^2 without bound; the "stable" vanilla rule destroys both.

	unconstrained		vanilla diag		spec-compiled (ours)	
System	test MSE	worst viol.	test MSE	worst viol.	test MSE	worst viol.
harmonic osc.	4.0×10^{-5}	2.6×10^{-1}	1.1×10^{-3}	1.0×10^{0}	2.4×10^{-4}	2.1×10^{-10}
anisotropic osc.	1.3×10^{-2}	7.0×10^{0}	2.0×10^{-2}	1.0×10^{0}	1.5×10^{-2}	3.1×10^{-13}
damped osc.	8.9×10^{-4}	5.8×10^{-1}	2.1×10^{-3}	0	8.2×10^{-4}	0
advection (per.)	2.8×10^{-10}	5.8×10^{9}	9.3×10^{-4}	1.0×10^{0}	2.3×10^{-8}	1.6×10^{-13}
heat (per.)	4.6×10^{-7}	2.5×10^{16}	2.9×10^{-4}	1.0×10^{0}	4.3×10^{-9}	2.8×10^{-13}
heat (Dir.)	9.0×10^{-7}	1.2×10^{17}	2.7×10^{-4}	0	2.4×10^{-7}	0
adv.–diff. (per.)	4.1×10^{-7}	5.2×10^{15}	4.9×10^{-4}	1.0×10^{0}	7.5×10^{-9}	7.3×10^{-14}
wave (per.)	1.7×10^{-4}	2.5×10^{4}	3.8×10^{-3}	1.0×10^{0}	6.3×10^{-4}	5.0×10^{-13}
Schrödinger	1.0×10^{-4}	2.3×10^{7}	3.7×10^{-3}	1.0×10^{0}	1.7×10^{-4}	1.3×10^{-13}
consensus	9.1×10^{-5}	6.2×10^{1}	2.8×10^{-4}	1.0×10^{0}	6.2×10^{-6}	1.0×10^{-13}

Table 5: Downstream case study: held-out one-step test MSE vs. worst violated obligation metric ($E_{\text{cons}}/E_{\text{diss}}/E_{\text{symp}}$) over a 1000-step rollout. "0": no dissipation obligation ever violated ($E_{\text{diss}} = 0$ exactly).

the classes a typed, verifier-grounded design makes attributable.

8 Related Work

Autoformalization and verifier-grounded mathematics. Translating informal mathematics into machine-checkable form spans proof autoformalization (Wu et al., 2022; Azerbayev et al., 2023), sketch-then-verify proving (Jiang et al., 2023), benchmarks such as miniF2F (Zheng et al., 2022), and mathematical language processing broadly (Meadows and Freitas, 2023). Our target language is far lighter than a proof assistant's, but the contract is the same: outputs are graded by a checker, not by string overlap. Restricting to quadratic/linear invariants of linear flows buys complete decidability, counterexample witnesses, and—unusual for autoformalization—a *compiler* from verified statements to learnable model classes. Solver-verified modeling (Huang et al., 2025) and self-correcting simulation (Park et al., 2026) share the refinement pattern; we add typed obligations and obligation-level metrics.

Structure preservation in numerics and learning. That conservation laws are properties of the discrete operator's algebraic form, not of an optimization target, is classical: discrete variational derivative methods (Furihata, 1999; Furihata and Matsuo, 2010; Yaguchi et al., 2010), variational integrators (Marsden and West, 2001), and their neural inheritors (Greydanus et al., 2019; Lutter et al., 2019; Chen et al., 2020; Matsubara and Yaguchi, 2023; Tanaka et al., 2025; Khosrovian et al., 2025); soft-constrained alternatives fail in characterizable ways (Raissi et al., 2019; Krishnapriyan et al., 2021; Donti et al., 2021). For the S4/S5/Mamba family (Gu et al., 2022; Smith et al., 2023; Gu and Dao, 2024; Dao and Gu, 2024) the generator is an explicit matrix, which is what makes specification-to-parameterization compilation exact. Our contribution is the constraint-*composition* view: obligations, not classes, as the unit of structure, including simultaneous conservation–dissipation (Proposition 2).

9 Conclusion

Mathematical text about dynamical systems denotes conjunctions of proof obligations, not single structure labels. We made that observation operational—a typed specification language with decidable atoms, an executable gold standard, a verifier that refutes with witnesses and refines with typed feedback, and a compiler from verified specifications to generators that satisfy them for every

parameter value. Upstream, frontier LLMs recover most of the conjunction once the answer space can express it, and one verifier round makes their assertions essentially always true; downstream, one-step accuracy says nothing about thousand-step structure, while compiled obligations survive by construction. We hope INVARIANTSPEC serves as a small, fully checkable testbed at the interface of mathematical language and verified computation.

Limitations

Our systems are low-dimensional, autonomous, and linear (by design: the two nonlinear systems exist to be refused). Selective-scan, input-dependent SSM parameterizations are outside the present formal target; extending the obligation language to time-varying generators $A(x_t)$—where the natural conditions become pointwise or averaged—is the main open direction. The benchmark's 12 systems support cluster-bootstrap intervals but not fine-grained per-family comparisons, and its linguistic variants are author-written rather than corpus-mined, so lexical diversity is bounded by design choices we controlled for (register, order, polarity) rather than by natural variation. The DSL certifies conservation of linear quantities and conservation/dissipation of quadratic ones, but cannot certify decay of a linear functional (there is no semidefinite witness for $|c^\top h|$ shrinking); models bump into this boundary on Dirichlet systems (§7), and extending the language with weighted or squared surrogates is straightforward future work. The refinement loop requires the system's executable discretization, which exists here by construction but must be synthesized (e.g. from the equation string) in fully open-ended settings; the loop runs for a single round with a single verifier, the minimal instance of verifier-guided repair. Finally, the evaluated models are accessed through a commercial gateway whose serving configuration we do not control, so absolute numbers should be read as a snapshot.

References

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. 2023. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*.

Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou. 2020. Symplectic recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Tri Dao and Albert Gu. 2024. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*.

Priya L. Donti, David Rolnick, and J. Zico Kolter. 2021. DC3: A learning method for optimization with hard constraints. In *International Conference on Learning Representations (ICLR)*.

Daisuke Furihata. 1999. Finite difference schemes for $\partial u/\partial t = (\partial/\partial x)^\alpha\, \delta G/\delta u$ that inherit energy conservation or dissipation property. *Journal of Computational Physics*, 156(1):181–205.

Daisuke Furihata and Takayasu Matsuo. 2010. *Discrete Variational Derivative Method: A Structure-Preserving Numerical Method for Partial Differential Equations*. Numerical Analysis and Scientific Computing Series. Chapman & Hall/CRC.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. 2019. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling (COLM)*.

Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations (ICLR)*.

Xuhan Huang, Qingning Shen, Yan Hu, Anningzhe Gao, and Benyou Wang. 2025. LLMs for mathematical modeling: Towards bridging the gap between natural and mathematical languages. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2678–2710.

Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. 2023. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *International Conference on Learning Representations (ICLR)*.

Razmik Arman Khosrovian, Takaharu Yaguchi, Hiroaki Yoshimura, and Takashi Matsubara. 2025. Poisson-Dirac neural networks for modeling coupled dynamical systems across domains. In *International Conference on Learning Representations (ICLR)*.

Aditi S. Krishnapriyan, Amir Gholami, Shandian Zhe, Robert M. Kirby, and Michael W. Mahoney. 2021. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Michael Lutter, Christian Ritter, and Jan Peters. 2019. Deep Lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations (ICLR)*.

Jerrold E. Marsden and Matthew West. 2001. Discrete mechanics and variational integrators. *Acta Numerica*, 10:357–514.

Takashi Matsubara and Takaharu Yaguchi. 2023. FINDE: Neural differential equations for finding and preserving invariant quantities. In *International Conference on Learning Representations (ICLR)*.

Jordan Meadows and André Freitas. 2023. Introduction to mathematical language processing: Informal proofs, word problems, and supporting tasks. *Transactions of the Association for Computational Linguistics*, 11:1162–1184.

Donggeun Park, Hyeonbin Moon, and Seunghwa Ryu. 2026. A self-correcting multi-agent LLM framework for language-based physics simulation and explanation. *npj Artificial Intelligence*, 2:10.

Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. 2023. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations (ICLR)*.

Yusuke Tanaka, Takaharu Yaguchi, Tomoharu Iwata, and Naonori Ueda. 2025. Energy-consistent neural operators for Hamiltonian and dissipative partial differential equations. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Takaharu Yaguchi, Takayasu Matsuo, and Masaaki Sugihara. 2010. An extension of the discrete variational method to nonuniform grids. *Journal of Computational Physics*, 229(11):4382–4423.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. miniF2F: A cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations (ICLR)*.

A The Exchange Format

Gold specification of periodic advection–diffusion, in the JSON exchange format produced by the models (schema-constrained):

```
{
  "quantities": [
    {"name": "mass", "type": "linear",
     "behavior": "conserved", "form": "mass"},
    {"name": "L2 norm", "type": "quadratic",
     "behavior": "dissipated", "form": "l2_norm"}
  ],
  "assumptions": ["periodic boundary conditions"],
  "proof_obligations": ["A^T c = 0", "A^T + A <= 0"],
  "representability": "exact_linear"
}
```

The verifier resolves `form` names against the system's discretization ($c = \mathbf{1}$, $M = I$ on the grid), checks $A^\top c = 0$ and $A^\top + A \preceq 0$, and transports both to the discrete map ($\Phi^\top c = c$, $\Phi^\top \Phi \preceq I$, Theorem 1).

B Verifier Details

Equalities are accepted at relative residual $< 10^{-10}$, inequalities at $\lambda_{\max} \leq 10^{-10}$ (scaled); all discrete certificates hold at $< 10^{-12}$ for the gold specifications. For failed equalities the witness is the extremal eigenvector v of the symmetrized residual $\mathrm{sym}(A^\top M + MA)$, reported with $\frac{d}{dt}\frac{1}{2}v^\top M v$; for failed inequalities it is the eigenvector of the largest positive eigenvalue. Systems of dimension ≤ 8 with near-rational entries are re-checked in exact arithmetic (SymPy); the float and exact verdicts agreed in every instance.

C Training Details

Teacher trajectories use the exact flow $\Phi_\Delta = \exp(\Delta A)$ (float64), 32 training / 8 held-out initial conditions, 80 steps, $\Delta = 0.05$; smooth random Fourier fields (modes ≤ 4) for PDE systems, standard normal states otherwise. All parameterizations train with Adam (lr 5×10^{-3}, 200 epochs, full batch) on one-step MSE through `matrix_exp`. Rollouts are 1000 steps. State dimensions: oscillators 2–4, graph consensus 8, PDE systems 24 (12×2 for wave/Schrödinger). The LLM evaluation uses temperature 0, low reasoning effort, `max_tokens`=4096, and OpenRouter structured outputs; the complete evaluation (4 models × 3 methods × 180 inputs, plus refinement calls) cost under $15.

D Additional Results

Figures 4 and 5 disaggregate Table 3 by method and by base system.

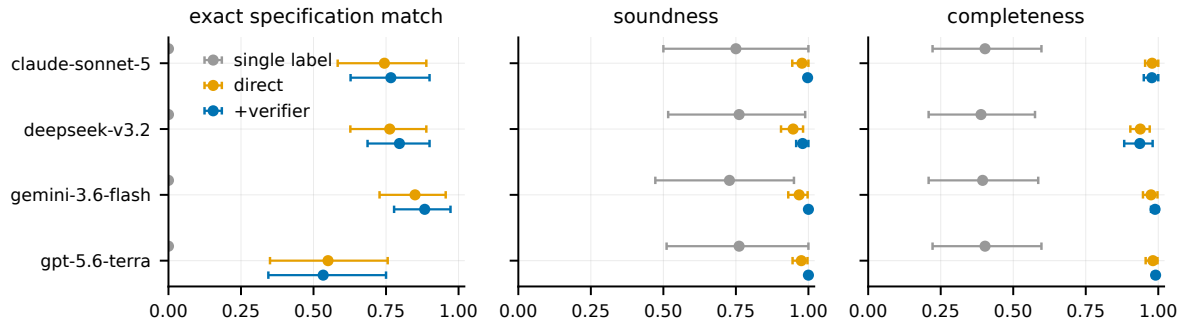


Figure 4: Exact specification match, soundness, and completeness per model and method, with cluster-bootstrap 95% CIs (gray: single label; orange: direct; blue: +verifier). The single-label baseline's exact match is pinned at 0 by its answer space; soundness and completeness show the same ceiling pattern across all four models.

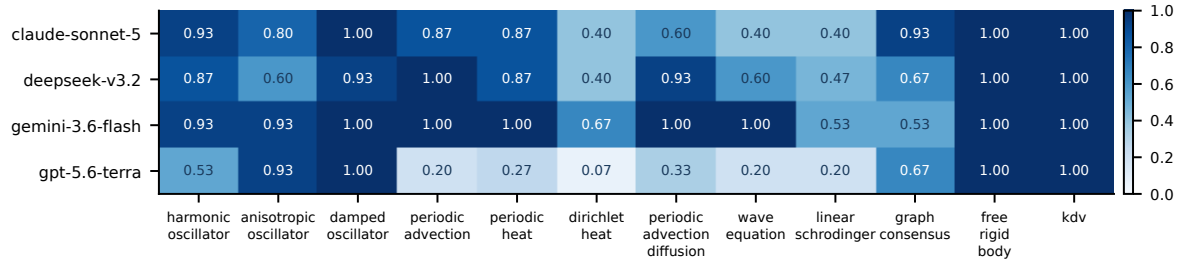


Figure 5: Exact specification match per base system (verifier-refined method; average over 5 variants $\times$ 3 modalities; parse failures count as 0). The hardest cells are systems whose specification validity depends on stated side conditions (periodic/Dirichlet boundary systems) rather than on obligation algebra.

E Benchmark Construction

Each system's five text variants are written to be system-identifying under text-only presentation (register varies: textbook, applied, terse, narrative, statement-style). The oracle control (§3) feeds gold specifications through the full scoring pipeline and attains 1.000 on every metric, confirming that fingerprinting, semantic matching, and aggregation introduce no noise floor. A seeded noisy oracle with programmed corruption rates (collapse, polarity, dropped assumptions, representability flips) reproduces the intended metric decreases, validating metric sensitivity.

MathQueue: Reasoning Scheduling Helps Only Within a Narrow Budget, and Answer Extraction Decides Whether It Helps at All

Jonghyun Lee
Artificial Intelligence
Graduate School of Engineering
Yonsei University
devjh@yonsei.ac.kr

Byungjun Kim
Research & Administrative
Information Team
Yonsei University
kbjoz87@yonsei.ac.kr

Abstract

Chain-of-Thought (CoT) prompting treats every reasoning step as equally important, even though solving a mathematical problem depends largely on deciding which step to take first. We therefore formulate reasoning as a scheduling problem. **MathQueue** decomposes a problem into tasks, scores each task by how much subsequent work it enables, how much information it yields, and how likely it is to introduce an error, and then solves the tasks in that order. We evaluate this formulation on GSM8K with two models under a fixed limit on the number of tokens the model may generate, and find that it helps over a far narrower range of conditions than anticipated.

Executing each task in a separate model call, with a verifier that may trigger replanning, performs substantially worse than standard CoT (83% to 72%) at three times the generated tokens and, once the re-sent context of twelve calls is counted, thirteen times the total; the replanning loop accounts for most of the loss. A single-pass variant performs better, but for reasons unrelated to prioritisation: it terminates in the required answer format on every problem, whereas CoT does so on only 87%, and it generates a third fewer tokens with a quarter of the variance in length, yet conditional on an answer being delivered it is never more accurate than CoT.

Retaining this structure but suppressing the ranking leaves accuracy unchanged at loose budgets and costs 19 points at the single budget tight enough to make ordering consequential. Intervening on the scheduler directly rather than on the prompt then locates that effect: the task graphs the model generates are near-chains, so our priority rule reorders only 9 of 100 problems, and the gain therefore comes from committing to a ranked plan rather than from the particular scores we assign to tasks.

Whether any of this constitutes an improvement depends on a factor we had not expected to matter: how the answer is parsed from a truncated generation. Scoring only well-formed answers favours MathQueue in all eight comparisons; reading the last number instead reverses six of them. We derive an identity explaining the reversal, adjudicate between the rules by manually inspecting every non-compliant generation, and confirm on a second model that what STRICT scoring rewards is the baseline's format compliance rather than its reasoning. Budget-constrained evaluation of mathematical reasoning must therefore fix and report how the answer is read.

1 Introduction

Large language models solve mathematical word problems by generating a linear chain of intermediate steps (Wei et al., 2022; Nye et al., 2021). This makes an implicit assumption: every step deserves the same treatment. Human problem solving does not work this way. Expert solvers categorise a problem by the principle that governs it, whereas novices attend to surface features (Chi et al., 1981); recognising the governing relation is a high-value step, and multiplying two numbers afterwards is not. When the generation budget is finite, spending it in the wrong order is costly.

A large body of work adds structure to address this: search over branches and thought graphs (Yao et al., 2023a; Besta et al., 2024), sample and vote (Wang et al., 2023), decompose into sub-questions (Zhou et al., 2023; Khot et al., 2023), interleave reasoning with external actions (Yao et al., 2023b), or refine a draft with model-generated feedback (Madaan et al., 2023). All of them cost more than a single CoT pass, and several measure that cost carefully: Wang et al. (2023) equalise sample counts, Yao et al. (2023a) compare token consumption against 100 independent CoT trials, and Besta et al. (2024) fix the number of thoughts to equate cost. In every case, however, the budget is matched *post hoc* rather than imposed at decode time as a

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 112–125
October 29, 2026 ©2026 Association for Computational Linguistics

constraint the method must plan against, and no method is scored on what it has produced when the budget is exhausted.

We take the scheduling view literally: a solver should enumerate candidate reasoning tasks, estimate each task's priority, and execute in priority order subject to dependencies. We then ask, under a controlled token budget, whether this helps.

The answer depends on a decision we had not initially considered part of the method at all: how the final answer is read from a possibly truncated generation. Our contributions are:

1. We cast mathematical reasoning as priority-scheduled traversal of a reasoning-task directed acyclic graph (DAG) (§3), with an interpretable priority functional and a balance operator over reasoning types.

2. We report a **negative result**: a multi-turn reasoning queue with verification and dynamic replanning degrades GSM8K accuracy by 11 points at $\approx 3\times$ the generated and $\approx 13\times$ the total token cost, and an ablation localises the loss in the replanning loop (§6.1).

3. We identify a **plan-prefix floor**: single-pass scheduling spends $\Lambda \approx 127$ median tokens stating its plan before solving, so below $B \approx \Lambda$ it collapses to near-zero accuracy under every scoring rule (§6.3).

4. We separate **ordering from structure** with an ablation that plans without ranking, and find that ordering is beneficial only within the window in which the budget binds and the plan fits (§6.5). Intervening on Eq. 5 directly then **localises** that benefit: because generated task graphs are near-chains, it derives from committing to a ranked plan rather than from our particular task scores (Appendix F).

5. We show that for $B \geq 256$ the **extraction rule can determine the sign** of the comparison, reversing six of eight comparisons, and give an exact identity explaining why (§4, §6.4). We then **adjudicate** between the candidate rules by manual inspection of every non-compliant generation, which yields a validated truncation-aware extractor (Appendix D).

6. We establish **what the advantage consists of**: on a second model whose CoT baseline already complies on 99% of problems it disappears and all three extractors agree, confirming that STRICT-scored gains track format compliance; and **budget-matched self-consistency** is either far worse than MathQueue or equal to it at $3.8\times$ the generated tokens (Appendices G and H).

2 Related Work

Structured prompting. CoT (Wei et al., 2022) and scratchpads (Nye et al., 2021) externalise intermediate computation. Least-to-Most (Zhou et al., 2023) identifies all sub-problems in one pass and solves them in the easy-to-hard order fixed at decomposition time; Decomposed Prompting (Khot et al., 2023) relaxes this to recursive and other non-linear structures, generating each sub-question conditioned on previous answers. Both therefore order sub-problems by a rule, but neither rule refers to a generation budget. Kim et al. (2024) go further and dispatch a planned task DAG in parallel, optimising latency rather than what is produced under a token cap. In our notation these methods realise a traversal of the dependency graph that is unordered with respect to any explicit cost, whereas MathQueue selects the linear extension that is greedy with respect to a stated priority functional.

Search and self-correction. Tree- and graph-structured search (Yao et al., 2023a; Besta et al., 2024) and self-consistency (Wang et al., 2023) improve solution quality by exploring several candidate paths, at a token cost that is a multiple of a single chain. Self-Refine (Madaan et al., 2023) adds iterative feedback from the model itself, but Huang et al. (2024) show that in the *intrinsic* setting, where no external feedback is available, accuracy on reasoning benchmarks typically *degrades* rather than improves, and that previously reported gains rely on oracle labels. Our verifier is intrinsic in exactly this sense, and our negative result is consistent with theirs: on the problems where it triggers replanning, final accuracy falls below what the same pipeline achieves with the loop disabled.

Efficient reasoning. Recent work studies the cost side of reasoning explicitly, by shortening outputs (Nayab et al., 2026) or allocating test-time compute (Snell et al., 2025); see Sui et al. (2025) for a survey. We contribute a budget-matched comparison in which the manipulated variable is the *order* of reasoning rather than its length or sampling count.

Answer extraction. Evaluating GSM8K-style benchmarks requires parsing a numeric answer out of free-form text. GSM8K marks its gold answers with a #### delimiter and its reference implementation credits a completion only when that marker is present (Cobbe et al., 2021), which is the rule we call STRICT. Downstream practice relaxes it: Kojima et al. (2022) fall back to the last number when the answer trigger is absent, and the widely used harness of Biderman et al. (2024) offers both filters as interchangeable options, treating the choice as a neutral implementation detail. Scores and even model rankings are known to be sensitive to such surface choices (Sclar et al., 2024; Biderman et al., 2024). We identify a specific mechanism behind these reversals: once generations are truncated, the fallback systematically credits whichever method failed to deliver an answer.

3 MathQueue

3.1 Budget-constrained solving

Let $q \in \mathcal{Q}$ be a problem with gold answer $y^\star(q)$, and let π_θ be an autoregressive model over token alphabet Σ. A *solver* produces a completion $u \in \Sigma^*$ from which an extractor recovers a prediction. Decoding is greedy ($T = 0$) under a completion budget $B \in \mathbb{N}$, so the realised output is the prefix $u_{1:\min(|u|,B)}$.

3.2 Reasoning task graph

Given q, the model emits reasoning tasks $\mathcal{T} = \{t_1, \dots, t_k\}$ with $3 \le k \le 6$ and a dependency relation $E \subseteq \mathcal{T} \times \mathcal{T}$, forming a directed graph $G = (\mathcal{T}, E)$ with parents $\mathrm{pa}(t_i) = \{t_j : (t_j, t_i) \in E\}$. Each task carries a type $\tau(t_i)$ drawn from $\mathcal{K} = \{\text{PLN}, \text{CAL}, \text{VER}, \text{OTH}\}$ and a self-estimated feature vector

$$\phi(t_i) = (D_i,\, I_i,\, C_i,\, R_i)^\top \in [0,1]^4, \qquad (1)$$

where D is dependency (downstream work enabled), I is information gain, C is complexity, and R is error risk. Malformed output triggers one repair turn; persistent failure falls back to a fixed four-task chain.

3.3 Priority functional

Tasks are scored by a linear functional with weights $w = (\alpha, \beta, \gamma, \delta)^\top$:

$$P(t_i) = w^\top \phi(t_i) = \alpha D_i + \beta I_i + \gamma C_i + \delta R_i. \quad (2)$$

We fix $w = (0.35, 0.35, 0.10, 0.20)^\top$ a priori; since these are non-negative and sum to one, $P(t_i) \in [0,1]$. Dependency and information gain dominate so that structure-revealing steps precede execution steps, while complexity is down-weighted so that a long calculation cannot monopolise the queue merely by being long.

3.4 Balance operator

To prevent trajectories consisting only of arithmetic, we track the type distribution of the completed set $\mathcal{D}_s$ after s steps,

$$\rho_s(\kappa) = \frac{\big|\{t \in \mathcal{D}_s : \tau(t) = \kappa\}\big|}{|\mathcal{D}_s|}, \qquad \kappa \in \mathcal{K}, \qquad (3)$$

and, writing $n_s = |\mathcal{D}_s|$, add a bonus to under-represented types:

$$b_s(t_i) = \begin{cases} \lambda_1 & \tau(t_i) = \text{VER},\ n_s \ge 2, \\ & \rho_s(\text{CAL}) > \theta_c \ \vee\ \rho_s(\text{VER}) < \theta_v, \\ \lambda_2 & \tau(t_i) = \text{PLN},\ n_s \ge 3, \\ & \rho_s(\text{PLN}) < \theta_p, \\ 0 & \text{otherwise}, \end{cases} \qquad (4)$$

with $\lambda_1 = 0.25$, $\lambda_2 = 0.20$, $\theta_c = 0.55$, $\theta_v = 0.15$, $\theta_p = 0.10$. When both clauses apply the maximum is taken, so $P(t_i) + b_s(t_i) \in [0, 1 + \lambda_1]$. This is a reasoning-domain analogue of load balancing across resource classes.

3.5 Selection rule

Let $\mathcal{F}_s = \{t_i \in \mathcal{T} \setminus \mathcal{D}_s : \mathrm{pa}(t_i) \subseteq \mathcal{D}_s\}$ be the ready frontier. The scheduler executes

$$t^{(s)} = \arg\max_{t_i \in \mathcal{F}_s} \big[P(t_i) + b_s(t_i)\big], \qquad (5)$$

falling back to $\mathcal{T} \setminus \mathcal{D}_s$ when $\mathcal{F}_s = \varnothing$, and halts after $S_{\max} = 6$ steps or when the queue empties.

Proposition 1 (Schedule validity)**.** *If G is acyclic and every dependency reference resolves to a task in $\mathcal{T}$, then $\mathcal{F}_s \neq \varnothing$ whenever $\mathcal{D}_s \subsetneq \mathcal{T}$, and the sequence produced by Eq. 5 is a linear extension of G.*

Equation 5 is a list-scheduling rule in the sense of Graham (1966), and Proposition 1 restates a standard fact: a finite non-empty DAG restricted to $\mathcal{T} \setminus \mathcal{D}_s$ always contains a source, so greedy selection from the ready set yields a topological order (Kahn, 1962). We state it because the antecedent is consequential in practice: because task graphs are

model-generated, a cycle or an unresolved identifier triggers the fallback branch and the guarantee no longer holds, so Proposition 1 is conditional on well-formed output rather than an invariant of the system.

3.6 Instantiations

The same policy admits realisations with markedly different call complexity M, which is our main experimental variable.

MathQueue-Full (multi-turn). Each task is executed in its own call followed by a verifier call; if the verifier rejects, a replanning call inserts an alternative-strategy task. With S executed steps, V verifier calls and Π replanning calls, $M = 2 + S + V + \Pi$. Empirically $\bar{M} = 12.2$.

MathQueue-Lite (two-call). One call yields $(\mathcal{T}, E, \phi)$; the schedule is materialised offline by Eq. 5; a second call solves in that order. Here $M = 2$, and because $\mathcal{D}_0 = \varnothing$ the balance operator is inactive by Eq. 4.

MathQueue-Inline (single call). The policy is stated in the prompt and the model ranks and executes its own tasks in one generation, so $M = 1$.

4 Budget, Delivery, and Extraction

Our results depend on separating three factors that raw accuracy conflates: whether the reasoning is correct, whether an answer is *delivered* within the budget, and how the extractor behaves when it is not.

Definition 1 (Delivery length)**.** *For solver π and problem q, write $u = u^{\pi}(q)$ for the unconstrained completion. The delivery length $L^{\pi}(q)$ is the smallest ℓ such that $u_{1:\ell}$ contains a well-formed final-answer marker, and $L^{\pi}(q) = \infty$ if the generation never emits one.*

Assumption 1 (Prefix consistency)**.** *Under greedy decoding, imposing a budget truncates rather than alters the trajectory: $u^{\pi,B}(q) = u^{\pi,\infty}_{1:\min(|u|,B)}(q)$.*

We contrast three extractors applied to the *same* generation. STRICT accepts only a well-formed marker carrying a number. LENIENT prefers such a marker but otherwise falls back to the last number appearing anywhere in the text; this is the rule used by our pipeline and corresponds to the last-number filter of Biderman et al. (2024), whereas STRICT corresponds to their marker-only filter and to the reference implementation of Cobbe et al. (2021).

Neither rule is satisfactory, because each answers a different question: STRICT asks whether the solver delivered, LENIENT asks what number it last wrote. A generation can lack a marker for two reasons, and the fallback is trustworthy in only one. If it *ran to completion* without the required line, the solver has an answer and merely failed to format it, so its trailing number is that answer. If it was *cut off*, the trailing number is whatever quantity was being manipulated at truncation and does not constitute an answer. We therefore define a third rule, GATED, which accepts a marker and otherwise falls back to the last number only when the provider reports that generation stopped on its own. Appendix D validates the distinction manually: the fallback is correct on 36 of 36 completed generations and on only 62 of 381 truncated ones, so GATED retains the rescues that are justified and discards those that are incidental.

Under STRICT, a solver scores only when it both delivers within the budget and is correct, so writing $\mathcal{C}^{\pi}(q)$ for the event that the delivered answer is correct,

$$\mathrm{Acc}^{\pi}_{\text{STRICT}}(B) = \Pr_{q}\big[\,\mathcal{C}^{\pi}(q) \;\wedge\; L^{\pi}(q) \le B\,\big], \quad (6)$$

which is non-decreasing in B. STRICT is thus the product of a *delivery rate* $\Pr[L^{\pi} \le B]$ and a *compliant accuracy* $\Pr[\mathcal{C}^{\pi} \mid L^{\pi} \le B]$; reporting only the product lets a formatting advantage read as a reasoning advantage, so we report both factors (§6.5). LENIENT adds a *rescue term*: when no answer is delivered, the fallback may still return the gold value by chance, for instance because the running computation happened to end on it. Writing

$$\varepsilon^{\pi}(B) = \Pr_{q}\big[L^{\pi}(q) > B \wedge \text{fallback correct}\big] \ge 0, \quad (7)$$

we have $\mathrm{Acc}^{\pi}_{\text{LENIENT}}(B) = \mathrm{Acc}^{\pi}_{\text{STRICT}}(B) + \varepsilon^{\pi}(B)$, while GATED restricts that rescue term to untruncated generations,

$$\mathrm{Acc}^{\pi}_{\text{GATED}}(B) = \mathrm{Acc}^{\pi}_{\text{STRICT}}(B) + \varepsilon^{\pi}_{\text{fmt}}(B), \quad (8)$$

and therefore the following identity for the gap $\Delta(B)$ between two solvers.

Proposition 2 (Extraction identity)**.** *For any two solvers π_1, π_2, writing $\Delta_{(\cdot)}(B) = \mathrm{Acc}^{\pi_2}_{(\cdot)}(B) - \mathrm{Acc}^{\pi_1}_{(\cdot)}(B)$,*

$$\Delta_{\text{LENIENT}}(B) = \Delta_{\text{STRICT}}(B) - \big(\varepsilon^{\pi_1}(B) - \varepsilon^{\pi_2}(B)\big). \quad (9)$$

Equation 9 states that LENIENT scoring credits each solver in proportion to how often it *fails* to deliver. If one method is systematically more compliant, LENIENT scoring is biased against it by exactly the difference in rescue rates, and the measured sign may reverse. Crucially, ε depends on incidental properties of the text, including material echoed from the prompt, and is therefore not a stable property of the method.

MathQueue-Inline also pays a structural cost: it must state its ranked plan before solving, so its delivery length decomposes as $L^{\mathrm{MQ}} = \Lambda + R$, where Λ is the plan prefix. This imposes a floor: if $B < \Lambda$ then no answer is delivered and $\mathrm{Acc}^{\mathrm{MQ}}_{\mathrm{STRICT}}(B) = 0$, irrespective of the quality of the underlying reasoning.

5 Experimental Setup

Data and models. We sample $N = 100$ problems uniformly without replacement (seed 42) from the GSM8K test split (Cobbe et al., 2021) and use Solar Pro 3 (`solar-pro3`; hereafter Solar) through an OpenAI-compatible endpoint at temperature 0. All methods see identical problems. To test whether the effects are a property of one model we repeat the central comparison on `gpt-oss-20b` (OpenAI, 2025) served locally (Appendix G); that model streams a hidden reasoning channel whose tokens are charged against the same cap, so we log them separately.

Prompt versions. Both prompts originally illustrated the output format with a literal example, `FINAL_ANSWER: 42`. We found that truncated CoT generations re-quote this instruction, causing LENIENT to return 42 on 10 of 100 problems at $B = 256$. We therefore re-ran the entire sweep with the numeral removed. We report both versions, written LEGACY and CLEAN, because the difference between them is part of the finding. Appendix I summarises every prompt.

Conditions. **E1** compares CoT, MathQueue-Full, and MathQueue-Full without replanning at $B = 2048$. **E2** compares CoT and MathQueue-Inline at $B = 2048$. **E3** sweeps budgets 128, 256, 512, 1024 and 2048 for both methods and both prompt versions. **E4** re-scores every stored generation under all three extractors, so the extractor is isolated from the model. **E5** adds an ablation, PLAN, which keeps MathQueue-Inline's decompose–list–solve–verify structure and its final-answer format but is explicitly instructed not to rank or prioritise the tasks; it is a prompt-level manipulation, differing from MathQueue-Inline in the ranking instruction and from CoT in having any explicit structure at all. **E6** replaces that manipulation with an intervention on Eq. 5 itself: one decomposition per problem, two linear extensions of the same graph, identical solve prompt. **E7** repeats E2, E4 and E5 on the second model at $B \in \{256, 512, 2048\}$. **E8** adds a budget-matched self-consistency baseline (Wang et al., 2023) in which $k = 3$ samples are each capped at $\lfloor B/k \rfloor$ tokens, so the ensemble cannot outspend a single CoT pass of B. E6–E8 are reported in Appendices F–H.

Metrics. We report accuracy under each extractor, mean completion tokens $\bar{c}$, mean calls $\bar{M}$, latency, truncation rate, *marker rate*, the share of generations from which a well-formed answer can be read, and *compliant accuracy*, accuracy restricted to those generations. Appendix J lists the decoding settings and the contents of the per-problem logs.

Significance. Paired comparisons use the exact conditional McNemar test (McNemar, 1947; Dietterich, 1998). With discordance counts n_{10}, n_{01} and $m = n_{10} + n_{01}$, the two-sided p-value is

$$p = \min\left(1,\ 2^{1-m} \sum_{i=0}^{\min(n_{10},n_{01})} \binom{m}{i}\right). \quad (10)$$

6 Results

6.1 E1: agentic scheduling degrades accuracy

The fully agentic queue is harmful. At $B = 2048$ on the LEGACY prompt under LENIENT scoring, MathQueue-Full reaches 72.0 against CoT's 83.0, a loss of 11 accuracy points, while issuing $\bar{M} = 12.2$ calls per problem ($p = 0.019$). Removing dynamic replanning recovers 8 of the 11 lost points (80.0; against CoT $p = 0.61$), which localises most of the degradation in the verify-and-replan loop rather than in prioritisation.

The cost of the loop is larger than generated tokens alone suggest. MathQueue-Full produces $2.9\times$ as many completion tokens as CoT, but each of its 12.2 calls re-sends the accumulated context, so it consumes $35\times$ the *input* tokens (5859 against 167) and $12.6\times$ the total (7013 against 558). Under any pricing scheme that charges for input, the orchestration rather than the reasoning dominates the cost; Appendix E reports both sides for every condition.

The replanning subset makes this precise. Let $\mathcal{R} = \{q : \Pi(q) > 0\}$; here $|\mathcal{R}| = 16$ with 2.8 insertions on average. Restricted to $\mathcal{R}$, MathQueue-Full attains 43.8%, CoT 62.5%, and the identical pipeline with the loop disabled 68.8%. Because the no-replan variant differs only in whether the loop may be invoked, the verifier is not merely selecting difficult problems; it degrades trajectories that the same scheduler would otherwise have solved.

6.2 E2: single-pass scheduling is compact and compliant

Compiling the policy into one generation removes the orchestration cost. At $B = 2048$ on the CLEAN prompt, MathQueue-Inline uses 33.4% fewer completion tokens than CoT (269 against 404) and is markedly more predictable: the standard deviation of completion length falls from 339 to 84 tokens, a $4\times$ reduction, and the maximum from 2048 (i.e. reaching the cap) to 549 (Appendix B). It also delivers an answer far more reliably. Even at this budget, where only 2% of CoT generations are truncated, CoT emits a well-formed final answer on 87% of problems against MathQueue's 100%; the remainder terminate in commentary on the requested format without producing the required line. Accuracy is deferred to Table 1, since it depends on the extractor.

Prompt length partly offsets the saving: the scheduling instructions lengthen the prompt from 165 to 281 tokens, so the reduction in *total* tokens is only 3.3% ($569 \to 550$). The saving is concentrated on the generated side, which dominates latency and, under most pricing schemes, cost.

6.3 E3: the plan-prefix floor

Sweeping the budget reveals a cost that the unconstrained comparison hides. MathQueue-Inline must state its ranked plan before solving; we locate the plan/solution boundary in all 100 traces and estimate Λ at a median of 127 tokens (mean 141, IQR 111–151). At $B = 128$ the plan alone consumes the entire budget on 50% of problems, and the plan-prefix floor becomes binding: MathQueue reaches 0% (STRICT) or 5% (LENIENT) against CoT's 7% and 27%. This is the one regime in which both scoring rules agree that scheduling is harmful, and it constitutes a substantive limitation of the method: the scheduling overhead must be amortised before ordering can be of any benefit.

	STRICT			GATED			LENIENT		
B	CoT	MQ	Δ	CoT	MQ	Δ	CoT	MQ	Δ
128	**7**	0	-7	**8**	0	-8	**27**	5	-22
256	42	**51**	$+9$	49	**51**	$+2$	**66**	54	-12
512	69	**83**	$+14$	78	**83**	$+5$	83	**84**	$+1$
1024	78	**85**	$+7$	**86**	85	-1	**87**	85	-2
2048	77	**84**	$+7$	**88**	84	-4	**88**	84	-4

Table 1: E3/E4 accuracy (%) on the CLEAN prompt under the three extractors. STRICT and LENIENT disagree on the sign at three of the four budgets with $B \geq 256$; under the validated GATED rule the gap is within 5 points of zero throughout and never significant ($p \geq 0.27$). LEGACY results are in Appendix A and the corresponding curves in Appendix B.

6.4 E4: extraction decides the sign

For $B \geq 256$ the conclusion depends entirely on how the answer is read (Table 1). Under STRICT MathQueue leads at every budget in this range, in all eight comparisons across both prompt versions, by $+3$ to $+14$ points; under LENIENT it trails in most, by as much as -12. The two rules disagree on the sign at six of those eight combinations, agreeing only where the LENIENT margin is within 6 points of zero.

Which rule to prefer can be settled empirically rather than by convention, because each makes a verifiable claim about individual generations. We manually inspected every generation carrying no marker and asked whether its trailing number is nonetheless the answer the solver had reached: pooled over budgets and methods, the fallback is correct on 36 of 36 generations that ran to completion and on 62 of 381 that were truncated (Appendix D). Under the resulting GATED rule of Eq. 8, MathQueue's advantage falls from between $+7$ and $+14$ points to between -4 and $+5$, and is not significant at any budget. Most of what STRICT credited to scheduling was CoT being penalised for correct answers in the wrong format.

Proposition 2 accounts for this exactly. Taking π_1 to be CoT and π_2 MathQueue, so that Δ is signed as in Table 1: at $B = 2048$ on the CLEAN prompt, $\varepsilon^{\mathrm{CoT}} = 88 - 77 = 11$ points while $\varepsilon^{\mathrm{MQ}} = 84 - 84 = 0$, so Eq. 9 gives $\Delta_{\text{LENIENT}} = +7 - (11 - 0) = -4$, which is the value we measure. The rescue term is large for CoT and zero for MathQueue for a reason evident in the delivery rates (Appendix C): above $B = 256$ MathQueue delivers a marker on every problem, so the fallback has nothing to rescue, whereas CoT

plateaus at 87% even when it is not truncated. At $B = 256$ the rescued items can be counted directly: exactly 24 CoT generations deliver no answer yet have the gold value as their trailing number, matching $\varepsilon^{\mathrm{CoT}} = 66 - 42 = 24$ points, against 3 for MathQueue. Appendix D presents one such case, in which CoT computes the correct profit and then expends its remaining budget on commentary about the required output format instead of stating the answer.

The instability is not confined to the choice of extractor. Under LENIENT scoring at $B = 256$, deleting the numeral from the format example alone moves the gap from $+6.0$ to -12.0, an 18-point change produced by a single token of prompt text with the model, problems and method held fixed. Under STRICT scoring the same change moves the gap only from $+13.0$ to $+9.0$.

6.5 E5: ordering versus structure

MathQueue differs from CoT in two respects simultaneously: it imposes a plan, and it orders that plan by priority. The experiments reported so far therefore cannot attribute the effect to either component. Table 2 separates them.

At loose budgets the ordering policy contributes nothing. At $B = 2048$, PLAN reaches 85.0 against MathQueue's 84.0 ($p = 1.000$), and at $B = 512$ the gap is $+2.0$ in MathQueue's favour ($p = 0.754$). Both variants deliver a well-formed answer far more reliably than CoT (93–100% against 75–87%), and both exceed it under STRICT scoring, by $+7$ to $+14$ points. Any advantage that scheduling confers in the unconstrained regime is therefore attributable to *structure*, that is, to committing to a task list and terminating in the required format, rather than to the priority functional.

At $B = 256$ this relationship reverses. Ranking is worth $+19.0$ points over the same structure left unranked ($p = 0.003$), the largest and most significant effect we measure for the ordering policy, and it is the one effect in the paper that is invariant to the extraction rule: the same $+19.0$ holds under GATED ($p = 0.003$). Unranked planning is in fact *inferior* to standard CoT in this regime (-10.0 STRICT, -17.0 GATED): it incurs the cost of writing a plan and then executes it in an arbitrary order, so it is truncated on 66% of problems and delivers an answer on only 34%, against MathQueue's 44% and 57%.

This is precisely the window that the analysis of §4 predicts. Ordering can matter only when part of the trajectory will be truncated, which requires B to bind, and it can help only when the plan itself fits, which requires $B > \Lambda$. Outside $(\Lambda,\ \approx 512)$ the priority functional is inert. We treat this window as a preliminary estimate: it rests on one binding budget, one dataset and one seed, and its upper edge is bounded only between 256 and 512. Intervening on Eq. 5 directly, by reordering an identical generated graph rather than by altering the prompt, localises the mechanism further: the graphs the model produces are near-chains, so within this window ranking operates chiefly by committing the model to one ordered plan rather than by selecting among genuinely competing tasks (Appendix F).

B	Method	STRICT	GATED	LENIENT	Mark.	Compl.	$\bar{c}$
256	CoT	42.0	49.0	**66.0**	44	**95.5**	**223**
	PLAN	32.0	32.0	43.0	34	94.1	240
	MQ	**51.0**	**51.0**	54.0	**57**	89.5	229
512	CoT	69.0	78.0	83.0	75	**92.0**	317
	PLAN	81.0	81.0	81.0	93	87.1	300
	MQ	**83.0**	**83.0**	**84.0**	**98**	84.7	**268**
2048	CoT	77.0	**88.0**	**88.0**	87	**88.5**	404
	PLAN	**85.0**	85.0	85.0	**100**	85.0	300
	MQ	84.0	84.0	84.0	**100**	84.0	**269**

Table 2: E5 ablation (CLEAN prompt). PLAN keeps the structure but drops the ranking; Compl. is accuracy among generations that delivered a marker. Ranking is worth $+19.0$ points over unranked structure at $B = 256$ under both STRICT and GATED ($p = 0.003$), but $+2.0$ ($p = 0.754$) at 512 and -1.0 ($p = 1.000$) at 2048. MathQueue has the lowest compliant accuracy at every budget.

The compliant accuracies in Table 2 delimit what structure provides. At every budget MathQueue is the *least* accurate of the three methods conditional on having delivered an answer (84.0–89.5% against CoT's 88.5–95.5%); its STRICT advantage is produced entirely by delivering answers at all. Scheduling in this setting improves the reliability with which a solution is emitted, not the quality of the solution.

7 Analysis

Structure yields delivery; ordering yields time. The ablation separates the two effects cleanly. Structure is responsible for the robust, budget-independent gains: a 100% marker rate against CoT's 87%, 33% fewer completion tokens, and a $4\times$ tighter length distribution. The priority functional adds nothing to this once the budget is loose, and contributes substantially ($+19$ points) precisely

where the budget compels the solver to choose what to address first. Both components of the scheduling formulation are substantive, but they operate in different regimes, and reporting only unconstrained accuracy would have revealed neither.

Why lenient scoring inverts the ranking. A last-number fallback is a reasonable heuristic for a completed generation and a poor one for a truncated generation, in which the last number is whatever quantity the model was manipulating when the budget ran out. Manual inspection quantifies the gap: the fallback is correct on 36 of 36 completed non-compliant generations and on 62 of 381 truncated ones. By Eq. 9 the unconditional form of the heuristic therefore awards credit to exactly the method that more often fails to deliver, which is why we adopt the finish-conditioned rule GATED as our primary measure and report all three. STRICT is not free of judgement either, since it penalises correct reasoning that terminates in the wrong format; GATED separates the two failure modes that STRICT and LENIENT each conflate.

Verification is not free. The verifier was intended to implement the dynamic-replanning component of the scheduling formulation, and it is precisely the component that fails: on $\mathcal{R}$, disabling it raises accuracy from 43.8% to 68.8%. This is consistent with evidence that LLMs cannot reliably self-correct reasoning without external signals (Huang et al., 2024). A scheduler that reorders work is inexpensive; one that also adjudicates correctness inherits the verifier's error rate.

Evaluation under a budget. Comparisons that fix a generation budget are becoming common as test-time compute becomes a controlled variable, whether the budget is enforced by truncation or by intervening on the decoder (Snell et al., 2025; Muennighoff et al., 2025; Sui et al., 2025). Our results suggest they are not identifiable unless the answer-extraction rule is fixed and reported, because truncation makes the extractor a co-determinant of the score. We recommend reporting the marker rate and the compliant accuracy alongside accuracy, and conditioning any fallback on the finish reason. The second model supports the diagnosis: where the baseline already complies on 99% of problems, all three extractors agree to within a point and no method ordering survives (Appendix G).

8 Conclusion

We formalised mathematical reasoning as priority-scheduled traversal of a reasoning-task DAG and evaluated it under matched token budgets. The agentic realisation performs worst: -11 accuracy points at triple the generated and thirteen times the total token cost, with the replanning loop responsible for most of the loss. The single-pass realisation carries a fixed plan-prefix cost that causes it to collapse below $B \approx \Lambda$. Between that floor and the budget at which truncation ceases to bind, priority ordering is worth $+19$ points over the same plan left unranked under both scoring rules that apply there; outside that window it confers no benefit, and every remaining gain derives from structure, which raises delivery to 100% at a third fewer tokens without improving accuracy conditional on delivery. Whether any of this manifests as an accuracy improvement depends on the answer-extraction rule, which reverses the sign of six of eight comparisons and whose choice we settle by manual validation rather than by convention. The scope over which the scheduling formulation is useful is thus narrower than it initially appears, and the methodological implication is broader: budget-constrained evaluation of mathematical reasoning must fix and report how the answer is read.

Limitations

Our study is deliberately narrow. All experiments use one dataset (GSM8K); grade-school word problems may understate the value of scheduling on problems where strategy search is the difficulty, and we have not verified that these effects transfer to MATH (Hendrycks et al., 2021) or to proof-oriented benchmarks. The two models we use differ in scale, provider and reasoning interface simultaneously, so the second serves to test whether a mechanism reappears, not to establish a population of models. With $N = 100$ per condition, individual budget points are underpowered: only 4 of the 10 STRICT comparisons reach $p < 0.05$, and of the eight at $B \geq 256$ where MathQueue leads, only two do. Our claim of a consistent STRICT advantage therefore rests on the sign being positive in 8 of 8 comparisons rather than on any single test, and under GATED the corresponding advantage is within 5 points of zero. We use a single seed and greedy decoding for the main comparisons, so we cannot separate method effects from decoding variance.

Several measurements are indirect. The plan prefix Λ is estimated by locating a section header and converting a character offset into tokens using a per-example ratio, not by counting tokens directly. Assumption 1 may be violated by provider-side nondeterminism. Stored traces are capped at 4000 characters, which affected 11 generations whose STRICT scores are therefore lower bounds. E1 predates the prompt correction and can only be scored LENIENT, because MathQueue-Full's answer arrives through a separate finalisation call rather than a single generation; we did not re-run it under the CLEAN prompt, so its -11 point result is not directly comparable to Table 1.

The method itself is only partly validated. The features ϕ are self-reported by the model and never checked against ground-truth step importance, the weights w were fixed a priori and never tuned, and we did not ablate the balance operator, which is inactive in MQ-Lite by construction. The E5 ablation isolates ordering from structure at the prompt level, so PLAN and MathQueue-Inline differ slightly in prompt length (250 vs. 281 tokens) as well as in ranking; the graph-level intervention of Appendix F removes that confound but can only reorder the graphs the model happens to generate, which are near-chains, so it bounds the effect of the selection rule on *these* graphs rather than of graph-based scheduling in general. Both were run at three budgets, so the window boundary between 256 and 512 is coarse. Our search baseline is budget-matched self-consistency at a single budget split (Appendix H); tree search and verifier-guided decoding remain untested.

Ethics Statement

This work evaluates arithmetic reasoning on a public benchmark (GSM8K) and involves no human subjects, no personal data, and no user-facing generation. The principal risk we identify is over-reliance on truncated model reasoning: under a tight budget both methods emit confident answers with incomplete justification, and part of MathQueue's advantage is that it *always* produces a clean final answer, which may make incomplete reasoning appear more trustworthy than it is. Our finding that a last-number fallback can invert a method ranking also implies that published gains on truncated-generation benchmarks may partly reflect parsing heuristics; we release per-problem logs so that any reported number can be recomputed under an alternative rule. Cost figures come from provider-returned token counts.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sid Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, et al. 2024. Lessons from the trenches on reproducible evaluation of language models. *Computing Research Repository*, arXiv:2405.14782.

Michelene T. H. Chi, Paul J. Feltovich, and Robert Glaser. 1981. Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5(2):121–152.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Computing Research Repository*, arXiv:2110.14168.

Thomas G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923.

Ronald L. Graham. 1966. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.

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

Arthur B. Kahn. 1962. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *International Conference on Learning Representations*.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. 2024. An LLM compiler for parallel function calling. In *International Conference on Machine Learning*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20275–20321.

Sania Nayab, Giulio Rossolini, Marco Simoni, Andrea Saracino, Giorgio Buttazzo, Nicolamaria Manes, and Fabrizio Giacomelli. 2026. Concise thoughts: Impact of output length on LLM reasoning and cost. *Information Sciences*, 757:123939.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *Computing Research Repository*, arXiv:2112.00114.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card. *Computing Research Repository*, arXiv:2508.10925.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design. In *International Conference on Learning Representations*.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*.

A Legacy-prompt results

Table 3 reports the sweep on the LEGACY prompt, whose format example contained the numeral 42. Comparing with Table 1 isolates the effect of that single token: under STRICT scoring the gap changes by at most 8 points and never changes sign for $B \geq 256$, whereas under LENIENT scoring it changes by up to 18 points and changes sign at $B = 256$.

	STRICT			LENIENT		
B	CoT	MQ	Δ	CoT	MQ	Δ
128	**20.0**	0.0	−20.0	**28.0**	3.0	−25.0
256	51.0	**64.0**	+13.0	60.0	**66.0**	+6.0
512	75.0	**81.0**	+6.0	**83.0**	82.0	−1.0
1024	79.0	**82.0**	+3.0	**83.0**	82.0	−1.0
2048	77.0	**82.0**	+5.0	**84.0**	82.0	−2.0

Table 3: Accuracy (%) on the LEGACY prompt.

B Budget and length curves

Figure 2 plots accuracy against the budget under both boundary extractors, and Figure 1 the completion-length distributions at $B = 2048$.

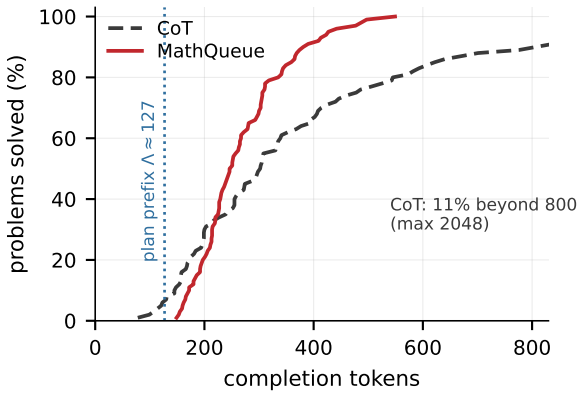


Figure 1: Completion lengths at $B = 2048$ (CLEAN). Scheduling concentrates generation into a narrow band and eliminates the long tail, at the price of a floor near the plan prefix Λ.

	marker %		trunc. %		$\bar{c}$	
B	CoT	MQ	CoT	MQ	CoT	MQ
128	8	0	95	100	127	128
256	44	**57**	61	**44**	223	229
512	75	**98**	20	**2**	317	**268**
1024	87	**100**	7	**0**	387	**265**
2048	87	**100**	2	**0**	404	**269**

Table 4: Delivery behaviour on the CLEAN prompt. Marker rate is the share of generations yielding a well-formed answer. Above $B = 256$ MathQueue always delivers, while CoT plateaus at 87% even without truncation.

C Delivery behaviour

Table 4 gives the quantities entering Eq. 9. Two features drive the results of §6.4. First, MathQueue's marker rate reaches 100% from $B = 1024$ upwards, so its rescue term ε is zero and its STRICT, GATED and LENIENT scores coincide. Second, CoT's marker rate stops at 87% even where only 2% of its generations are truncated, so its non-delivery is a formatting failure rather than a budget failure, and the rescue term stays large.

D Validating the GATED extractor

We examined every generation in the CLEAN sweep from which STRICT recovers nothing, and recorded whether the trailing number of the text is the value the solver had in fact arrived at. Table 5 reports the outcome. The split is close to categorical: when the model terminated of its own accord, the answer was present but unmarked in all 36 cases, whereas under truncation the trailing number was the gold value in 16% of cases, a rate consistent with a partial computation coinciding with the answer. The

Non-marker generations	n	correct	prec.
Terminated on its own	36	36	1.00
Truncated at the cap	381	62	0.16

Table 5: Manual validation of the last-number fallback, pooled over the five budgets and both methods on the CLEAN prompt. Precision is the share of non-marker generations whose trailing number is the gold answer. Conditioning on the finish reason separates a reliable rescue from an unreliable one, which is what Eq. 8 does.

composition of the non-marker cases follows the same pattern by budget: of CoT's 56 at $B = 256$, 49 were truncated and 24 carried the gold value; of its 13 at $B = 2048$, 11 did. MathQueue never produces a completed non-compliant generation at any budget.

Figure 3 shows a case that LENIENT rescues and GATED does not.

E Full cost accounting

Study	Method	in	out	total	$\bar{M}$	s
E1, 2048	CoT	167	391	558	1.0	2.9
	MQ-Full	5859	1154	7013	12.2	13.4
	no replan	5183	1021	6204	11.5	12.1
E5, 256	CoT	165	223	388	1.0	1.9
	PLAN	250	240	490	1.0	2.2
	MQ-Inline	281	229	510	1.0	2.1
E5, 2048	CoT	165	404	569	1.0	3.1
	PLAN	250	300	550	1.0	2.4
	MQ-Inline	281	269	550	1.0	2.8

Table 6: Mean prompt, completion and total tokens per problem, calls $\bar{M}$ and wall-clock seconds. The budget B caps completion tokens only, so input cost is unconstrained by construction, which is what makes the multi-turn realisation costly.

Table 6 reports both sides of the cost. The budget constrains completion tokens, so a method that issues many calls pays an input cost the budget does not see: MathQueue-Full re-sends the accumulated transcript on each of its 12.2 calls and consumes 35× CoT's input tokens. Single-pass scheduling avoids this entirely; its longer prompt adds a fixed 116 tokens, so at $B = 2048$ its total cost is comparable to CoT's (550 against 569) despite generating a third fewer tokens.

F E6: intervening on the selection rule

The E5 ablation manipulates the prompt, so it cannot distinguish the priority functional from the instruction to write a plan. We therefore intervene on

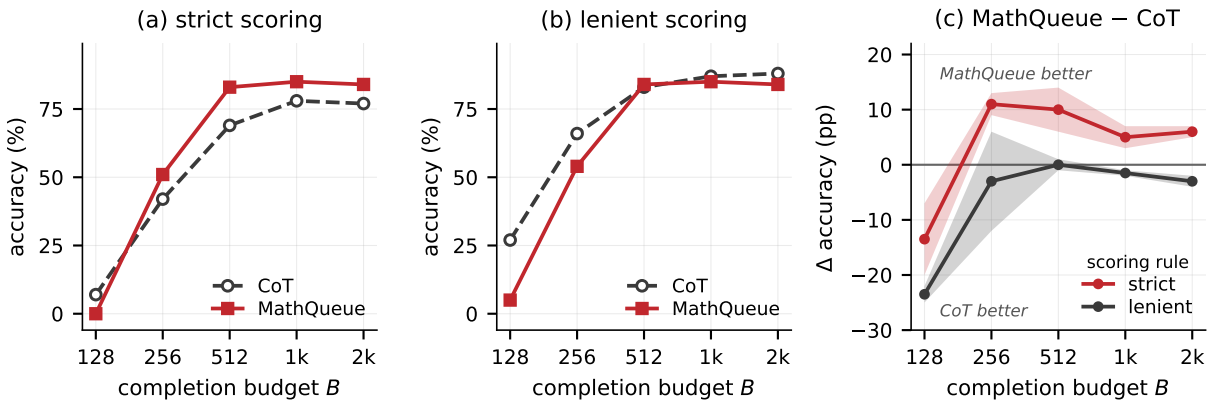


Figure 2: Accuracy against completion budget. (a) Under STRICT extraction MathQueue dominates CoT for $B \geq 256$. (b) Under LENIENT extraction the ordering reverses everywhere except at $B = 512$, where the two curves nearly coincide. (c) The gap $\Delta(B)$ for both prompt versions and both extractors: the extractor (colour) separates the curves cleanly, while the prompt version (solid vs. dotted) does not. At $B = 128$, where the plan prefix exceeds the budget, both rules agree that scheduling is harmful.

Problem (gold = 8): *Rose bought five dozens of eggs for $2.40 a dozen. She will sell it for $1 for 3 eggs. How much will be Rose's profit?*

CoT, truncated at $B = 256$: … *Profit = revenue − cost = $20 − $12 = $8.00. Thus profit = $8.00. But we need to show step-by-step reasoning. Also ensure final answer is numeric, maybe with dollar sign? The format says*

STRICT and GATED: nothing delivered ⇒ wrong.
LENIENT: trailing number 8.00 ⇒ correct.

MathQueue, complete in 242 tokens: … *4. Profit: $20.00 − $12.00 = $8.00*, followed by the required final-answer line.

All three rules: correct.

Figure 3: One of the 24 problems at $B = 256$ where LENIENT scoring rescues a CoT generation that never delivered an answer. Whether the rescue succeeds depends on which number the truncation happens to fall after.

Eq. 5 itself. For each problem we generate the task graph once and then execute it twice at $B = 512$, once in the order the priority functional selects and once in the order the model emitted, with an otherwise identical solve prompt from which all ranking language has been removed. Everything except the order is thus held fixed.

The intervention is possible on fewer problems than expected. Analysing the 100 graphs shows that 54 are chains admitting exactly one valid order, the median number of linear extensions over all graphs is 1, and the largest ready set encountered at any point is 3. The priority order coincides with the model's emitted order on 91 problems, and on 6 of the 9 remaining ones the disagreement is not attributable to the priority scores. Re-decomposition at execution time reproduced two genuinely distinct orders on 5 of the 9 pairs, and in every case they differ by a single transposition of adjacent tasks.

The outcome is unchanged on all of them. Both orders yield 44.4% STRICT and 55.6% GATED accuracy on the paired subset, the predictions agree token-for-token on the reordered pairs, and no problem changes from correct to incorrect or the reverse. This does not contradict §6.5; it locates the +19 points there. Within the window where the budget binds, what the ranking supplies is a commitment to a single ordered plan, and on graphs of this depth that commitment does not depend on which of the admissible orders is chosen. A benefit from the particular scores ϕ would require problems whose dependency structure leaves genuine choice, which GSM8K decompositions rarely do.

G E7: transfer to a second model

We repeat the E5 comparison on `gpt-oss-20b`, a model that differs from Solar in scale, provider and interface, and that streams a hidden reasoning channel whose tokens count against B (88 mean reasoning tokens for MathQueue against 58 for CoT at $B = 256$). Table 7 reports the result.

Two of the paper's claims reappear and one does not. The plan-prefix floor reappears, displaced upwards: because reasoning tokens are billed against the cap, the effective budget is smaller, and $B = 256$ now lies below the floor, where MathQueue reaches 25% against CoT's 66% and is truncated on 74% of problems. The value of ranking over

B	Method	STRICT	GATED	LENIENT	Mark.	Tr.
256	CoT	**66**	**66**	**72**	**73**	**27**
	PLAN	18	18	22	20	80
	MQ	25	25	32	26	74
512	CoT	**86**	**86**	**86**	**99**	**1**
	PLAN	**86**	**86**	**86**	95	5
	MQ	82	82	85	93	7
2048	CoT	85	85	85	**100**	**0**
	PLAN	**90**	**90**	**90**	**100**	**0**
	MQ	88	88	88	**100**	**0**

Table 7: E7: `gpt-oss-20b` on the same 100 problems and the CLEAN prompt. The baseline complies on 99–100% of problems from $B = 512$, so all three extractors agree and the STRICT advantage of §6.4 does not arise. Hidden reasoning tokens are charged against B, which moves the plan-prefix floor upwards.

unranked structure reappears in the same regime, at $+7$ points. What does not reappear is the STRICT advantage of §6.4, and the reason is the one our analysis predicts: this model's CoT baseline already delivers a well-formed answer on 99% of problems at $B = 512$ and 100% at $B = 2048$, leaving no compliance gap for structure to close. Consistently with that, all three extractors agree to within a point at every budget. The STRICT-scored advantage on Solar is therefore not a property of scheduling as such but of the compliance ceiling of the baseline it is measured against.

H E8: budget-matched self-consistency

B	Method	STRICT	GATED	Deliv.	out
512	SC (k=3)	45.0	48.0	51	484
	MQ	**83.0**	**83.0**	**98**	**268**
2048	SC (k=3)	84.0	84.0	93	1024
	MQ	84.0	84.0	**100**	**269**

Table 8: E8: self-consistency with $k = 3$ samples at temperature 0.7, each capped at $\lfloor B/k \rfloor$ tokens so that the total generation budget matches. Deliv. is the share of problems on which a majority answer could be formed; out is mean completion tokens.

Sampling several trajectories and taking a majority vote is the natural alternative use of a fixed budget. We match the budget by drawing $k = 3$ samples at temperature 0.7, each capped at $\lfloor B/k \rfloor$ tokens (170 at $B = 512$, 682 at $B = 2048$), and take the majority over delivered answers. Table 8 reports the comparison.

The two regimes separate sharply. At $B = 512$ the per-sample cap of 170 tokens is below the length of a typical solution, so most samples are truncated and no majority can be formed on 49% of problems; self-consistency reaches 45.0% against MathQueue's 83.0% ($p < 10^{-10}$). At $B = 2048$ each sample has sufficient room, and self-consistency matches MathQueue exactly (84.0% against 84.0%, $p = 1.000$) while generating $3.8\times$ as many tokens and taking $2.9\times$ as long. Within a tight budget, therefore, allocating tokens to one structured trajectory dominates spreading them over several unstructured ones, and once the budget is loose the two are equivalent in accuracy but not in cost. Neither regime provides evidence that sampling recovers what ordering provides.

I Prompts

Decomposition. The model returns JSON containing 3–6 tasks, each with `id`, `description`, `task_type`, `depends_on` and the four components of ϕ in $[0, 1]$, with instructions not to solve the problem yet and to include at least one VER task.

Inline scheduling. The single-call prompt instructs the model to (i) decompose the problem, (ii) rank tasks by dependency and information gain, stating explicitly that complexity must not dominate the order, (iii) solve in ranked order, (iv) verify once at the end, and (v) avoid long calculations before the governing relation is fixed. It requires a short ranked task list followed by the solution and a final answer line.

CoT baseline. The baseline asks for step-by-step reasoning and the same final-answer format, with no scheduling instructions. In the LEGACY version both prompts ended with the example `FINAL_ANSWER: 42`; in the CLEAN version the numeral is replaced by a placeholder, which the extractor ignores.

The paragraphs above summarise the prompts; the templates themselves are released verbatim as plain-text files alongside the logs, one per method, so that no wording has to be reconstructed from this description.

J Reproducibility

Models and endpoints. Solar Pro 3 is served as `solar-pro3` on the OpenAI-compatible endpoint `api.upstage.ai/v1`; the provider does not expose an immutable version tag, so every record carries the response identifier and timestamp returned by the API. The second model is `openai/`

`gpt-oss-20b` served locally behind an OpenAI-compatible endpoint, whose `reasoning_tokens` field is logged separately and counted against the same cap. Each run writes a `run_meta.json` recording the model string, endpoint, budget, seed, prompt version and library versions.

Decoding. Temperature is 0 for all main comparisons, at budgets of 128, 256, 512, 1024 and 2048 completion tokens, identical across methods within a study, with $S_{\max} = 6$ scheduling steps for multi-turn variants. Self-consistency (Appendix H) is the sole exception, at temperature 0.7 with $k = 3$. The budget is imposed as `max_tokens` at decode time rather than by post hoc truncation.

Problem sample. The 100 problems are drawn from the GSM8K test split by `random.Random(42).sample`, and the resulting indices are stored with the logs so that the sample can be reproduced or audited without re-running the sampler.

Records. Every problem yields one record containing the prediction, the gold answer, provider prompt/completion/total/reasoning token counts, wall-clock latency, the finish reason, the raw generation, and, for scheduling methods, the full task trace $(\mathcal{T}, E, \phi, \sigma)$ including the emitted order, the priority order and the executed order. Traces are stored uncapped for E6–E8; in the earlier sweeps they were truncated at 4000 characters, which affected 11 generations, as noted in the Limitations.

Offline scoring. All three extractors, the compliant accuracies and every p-value from Eq. 10 are recomputed from the stored generations by a single scoring script, so no number in this paper depends on scoring decisions made at generation time. This is what allows Table 1 to isolate the extractor from the model, and it allows any reader to re-derive the tables under a different extraction rule.

Typed Neuro-Symbolic Autoformalization of Quantitative Clinical Language: Executable Constraints and Selective Verification

Aleksandra Beliaeva[1,2,3]
[1]Applied AI Institute, Moscow, Russia
[2]SB AI Lab, Moscow, Russia
[3]BIMAI-Lab, University of Sharjah, Sharjah, United Arab Emirates

Abstract

Clinical eligibility criteria express numerical comparisons and Boolean relations in prose. We translate them into executable typed formulas using a neural parser, grammar and ontology constraints, and an independent symbolic verifier. The formulas have precise semantics, source-linked predicates, and an auditable record of type, unit, grounding, and consistency checks. On 200 independently double-annotated criteria, a development-selected threshold yields 89.2% logical correctness at 78.5% coverage. At matched 78% coverage, neuro-symbolic, neural, and rule selection achieve 89.7%, 88.5%, and 85.3% correctness, respectively. We evaluate these trade-offs with exact logical equivalence, confidence intervals, and full risk–coverage curves. Structural breakdowns, candidate oracles, augmentation experiments, and controlled edits across two backbones identify Boolean scope and omitted predicates as the main remaining errors. The symbolic interface makes these failures inspectable while separating formal validity from semantic correctness. The study provides an executable approach to quantitative semantic parsing and a controlled testbed for diagnosing its compositional bottlenecks.

1 Introduction

Autoformalization translates natural language into a formal language that can be checked and executed. We study quantitative clinical eligibility criteria. For example, "SBP > 190 or DBP > 95" combines two blood-pressure comparisons. A parser must recover both comparisons and preserve the word "or".

A simple scope error. Consider the intended formula and a possible parsing error:

$$\phi = (\text{SBP} > 190) \vee (\text{DBP} > 95),$$
$$\hat{\phi} = (\text{SBP} > 190) \wedge (\text{DBP} > 95).$$

For a record with SBP $= 200$ and DBP $= 90$ mmHg, ϕ is true and $\hat{\phi}$ is false. Both formulas use valid variables, units, and thresholds. Both can also point to the correct words in the source. Type checking and source-span checks therefore cannot detect this scope error. A comparison with the intended meaning is still necessary.

Our system separates these tasks. A neural parser proposes a Boolean abstract syntax tree (AST). During generation, token masks enforce a grammar and an ontology of supported variables and units. A separate verifier checks the completed AST and records each decision. A selector ranks candidates and can abstain. We test meaning independently by comparing each formula with an adjudicated reference on an exhaustive set of representative records.

The main finding is that Boolean scope and omitted predicates remain the principal bottlenecks. Candidate search, the tested augmentation schemes, and controlled edits on two backbones provide complementary evidence for this diagnosis. Typed formulas and exact evaluation make it possible to distinguish these semantic errors from malformed output, incompatible units, or invalid variables.

Selective prediction provides a practical operating point: the combined selector returns 89.2% logically correct formulas at 78.5% coverage. We compare systems at the same coverage and report full risk–coverage curves, so this accepted-set score can be assessed together with abstention. The symbolic layer also exposes each verification decision and its source evidence, making both accepted outputs and rejections auditable.

Contributions.

- We define a typed Boolean language for quantitative criteria and an exact test of logical equivalence within its supported fragment.
- We combine neural parsing, constrained gen-

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 126–135
October 29, 2026 ©2026 Association for Computational Linguistics

eration, symbolic checks, and selective prediction, and state what each component guarantees.

- We evaluate on 200 independently adjudicated criteria and analyze scope errors through structural breakdowns, candidate search, augmentation, and controlled edits across two backbones.

The implementation is available in our code repository. The ontology contains 15 quantitative variables. Our evaluation concerns their formal interpretation; it does not validate clinical decisions.

2 Task and Formal Semantics

Let x be a criterion and ϕ_x its interpretation. A leaf is $p = (v, o, a, u, n, s, e)$: ontology variable v, comparator o, typed value a, canonical unit u, negation flag n, inclusion/exclusion scope s, and source interval e. Internal nodes are AND, OR, and NOT; AND/OR have at least two ordered children and NOT one. Ranges compile to two conjunctive bounds. We retain children in source order. This preserves structure in the representation, but a parser can still omit a required leaf.

Let D_v be the typed domain of v, $\rho(v) \in D_v$ a complete record, and N_u canonical unit normalization. For a numeric leaf,

$$[\![p]\!]_\rho = I_{n,s}\big(o\big(N_u(\rho(v)), a\big)\big)\,,$$

where $I_{n,s}$ complements the truth value once if the leaf is negated or has exclusion scope, including when both conditions hold. Categorical equality is defined analogously. Boolean nodes have standard truth-functional denotations. Evidence e and confidence metadata are non-semantic: they are audited but do not change $[\![\phi]\!]$.

Serialization normal form. The primary JSON Schema permits at most three Boolean edges from the root and at most 12 children per AND/OR node. Variables and units are canonicalized, numeric spellings are normalized, ranges are expanded, and children preserve source order. Semantic AST exact match compares this ordered normal form while excluding evidence and confidence; the artifact hash covers the complete canonical JSON, including evidence. The auxiliary structure-first parser uses a separately reported depth-four, eight-child schema.

Checking meaning exactly. For a single comparison such as SBP > 190, it is enough to test one value below 190, the boundary 190, and one value above it. No comparison changes truth value inside any of these open intervals. We extend this idea to every variable and every threshold in a pair of formulas. For each variable, let T_v be the union of thresholds in the two formulas. The numeric cells are the singleton boundaries $\{t\}$ and open intervals between consecutive thresholds, including the two outer intervals. Finite categorical domains use singleton cells.

Lemma 1 (Threshold-cell completeness)**.** *For formulas over the supported scalar comparisons, canonical units, observed variables, and finite categorical domains, two formulas are logically equivalent iff they agree on the Cartesian product containing every categorical value, every numeric threshold, and one representative from every induced open cell.*

Proof. Every scalar comparison has constant truth value inside each induced cell. Structural induction over AND, OR, and NOT therefore makes each formula constant on every Cartesian-product cell. Agreement on one representative per cell is both necessary and sufficient. □

Thus our reported *logical equivalence* is exact whenever this Cartesian product is exhaustively enumerated; execution agreement is the fraction of its representatives on which denotations agree. Enumeration is exhaustive for all 200 human items and all 207 source-derived items.

3 Neuro-Symbolic Interface

Figure 1 shows the pipeline. Each leaf points to a source span. The verifier records which checks pass, warn, or fail.

3.1 Typed Neural Parser

We use Llama-3.1-8B-Instruct (Grattafiori et al., 2024) with deterministic decoding and a two-epoch LoRA trained on text-to-AST and verifier-derived contrastive records. The output budget is 768 tokens; generation stops after one complete JSON object. The grammar decoder compiles the bounded JSON Schema into token masks. Grammar masks exclude malformed JSON and invalid Boolean arity; ontology masks restrict variable names, value types, categorical domains, and

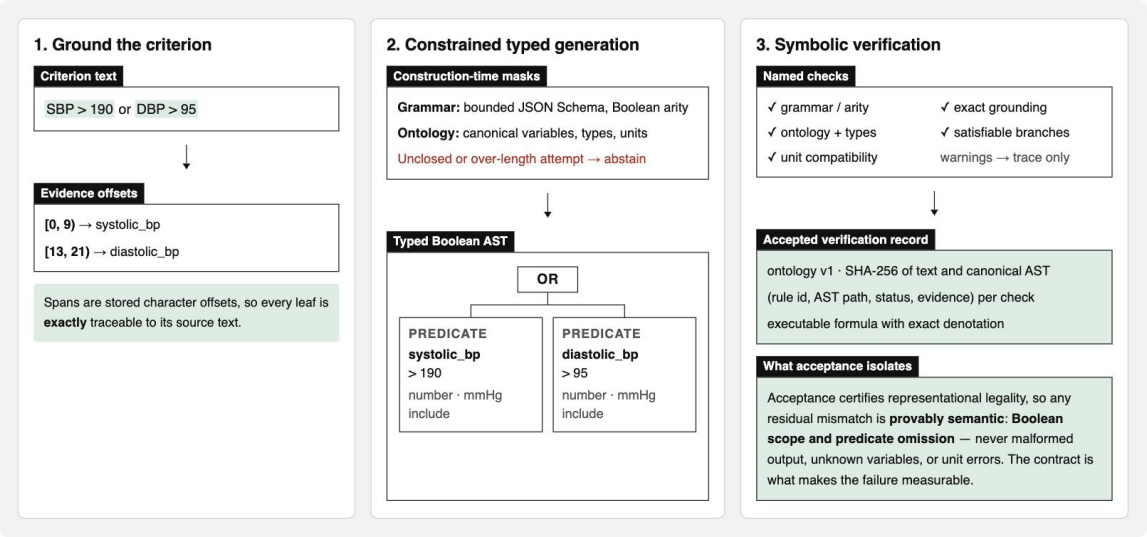


Figure 1: Typed autoformalization and verification. Source spans ground typed AST leaves; named checks expose grammar, ontology, unit, grounding, and consistency decisions. The record explains why a formula passes the implemented checks. It does not prove that the formula expresses the meaning of the text.

variable-compatible units. A length-capped or unclosed attempt abstains, so coverage exposes non-completion.

3.2 Ontology, Verification, and Selection

The declarative ontology contains canonical variables, aliases, value types, comparators, units, and weak plausibility ranges. For $\hat{\phi} = G_\theta(x)$, the verifier checks (i) grammar and arity, (ii) canonical variables, comparators, values, and units, (iii) exact evidence offsets and lexical support, (iv) plausibility, and (v) interval consistency in each conjunctive branch. A check is *decisive* exactly when its emitted status is `fail`; warnings remain in the trace but do not reject. The record stores the ontology version, hashes of x and the complete AST, and tuples $(\text{rule id}, \text{path}, \text{status}, \text{evidence})$.

What acceptance guarantees. An accepted output has valid Boolean arity and ontology-compatible variables, values, and units. Its evidence offsets point to text with lexical support for the leaves, and no implemented interval check finds a contradiction. The record binds these checks to the criterion and AST through hashes and paths. These checks establish formal validity under the implementation. They do not establish that every source predicate is present or that Boolean scope is correct, as the example in Section 1 illustrates. We measure those properties against adjudicated gold.

Compared systems. The *rule parser* has no learned parameters. It uses the released 15-variable alias lexicon, five comparator families, three range templates, two word-order patterns per comparator, and one negated-history pattern. Leaves are aligned to exact source spans; range bounds are joined by AND, and text bridges determine AND/OR composition. The implementation was fixed before final human adjudication. The *indexed parser* first generates a Boolean skeleton whose slots contain character offsets, then fills one typed predicate per slot; contiguous source-ordered slots are required, and its schema permits depth four and eight children. Both stages are trained only on the training split.

The *neural selector* selects from the neural candidate pool using verifier checks. Thus this baseline already uses symbolic verification; it is not an ablation that removes all symbolic components. The *neuro-symbolic selector* ranks one rule candidate, four neural candidates, and two indexed candidates. Its 23 features encode candidate source and index; verifier decision, score, checks, and warnings; evidence rate and mean generation log-probability; agreement with the rule and other neural candidates; and leaf count, depth, and root operator. A standardized ℓ_2-regularized logistic regression chooses $C \in \{.01, .1, 1, 10\}$ by out-of-fold development selection accuracy, breaking ties toward smaller C. Chia folds are grouped by NCT trial and Criteria2SQL by item. The sigmoid is the score

$r \in [0, 1]$; the threshold is the largest-coverage development point with observed precision at least .85 and a one-sided 95% Clopper–Pearson lower bound of .80. Model, C, and threshold are frozen before test evaluation.

For $g_\tau = \mathbf{1}[r \geq \tau]$, selector coverage is $c(\tau) = N^{-1} \sum_i g_\tau(i)$ and selective logical risk is $R(\tau) = \sum_i g_\tau(i) \mathbf{1}[\phi_i \not\equiv \hat{\phi}_i] / \sum_i g_\tau(i)$. We report the complete attainable risk–coverage curve. Parser coverage means completion of a valid AST; verifier coverage means a deterministic pass; selector coverage means final acceptance. The *Cov.* column in the central table always uses the last applicable decision.

Lemma 2 (Sound but incomplete interval rejection)**.** *After NOT elimination, consider an untruncated disjunctive-normal-form clause of canonical, non-negated numeric predicates using* $<, \leq, >, \geq, =$*. For each variable let* L *be the maximum asserted lower bound and* U *the minimum asserted upper bound, retaining endpoint inclusivity. If* $L > U$*, or* $L = U$ *with a strict endpoint, the clause is unsatisfiable. A disjunction is rejected as unsatisfiable only when every clause has such a conflict.*

Proof. Any satisfying value must lie in every asserted interval. The stated endpoint conditions make their intersection empty. A disjunction is unsatisfiable only if every disjunct is unsatisfiable. □

The check is incomplete: for example, $x = 0 \wedge x \neq 0$ is contradictory but is not detected by the implemented interval fragment.

4 Evaluation

We derive formulas from supported Chia relations (Kury et al., 2020) and Criteria2SQL conditions (Yu et al., 2020). Chia is grouped by NCT trial; Criteria2SQL official splits are preserved; normalized duplicate texts connect source groups. The release gate verifies zero normalized-text and NCT overlap and no official-test leakage (Appendix A). The resulting source rows number 981/318/442 for train/development/test; the deduplicated neural test has 207 criteria.

A blinded 200-item packet was independently annotated by two general-CS specialists under the closed formal guidelines, followed by third-person adjudication of every serialized disagreement. Annotators saw neither source-derived labels nor model outputs. Semantic AST agreement is .940, Boolean structure agreement .960, and root-operator $\kappa = .950$; exact span agreement is separately analyzed because span boundaries are non-semantic. The packet re-annotates 199 texts from the held-out evaluation pool plus one replacement, so it supplies independent gold rather than a newly sampled corpus.

We compare the rule, neural, indexed, and neuro-symbolic systems defined in Section 3. Metrics are normalized predicate F1, Boolean and full semantic-AST exact match, exact logical equivalence from Lemma 1, execution agreement, satisfiability match, type/unit violations, coverage, and accepted-set quality. Every execution comparison uses the same per-item universe, fixed before system comparison as the union of gold and all candidates in the declared pool; all 200 universes are exhaustively enumerated (at most 243 records). Overall metrics count abstentions as errors; accepted metrics condition on coverage. Exact binomial intervals are labelled *exact*; every bootstrap interval uses 10,000 resamples, clustered by NCT trial for Chia and by item for Criteria2SQL. Of the 200 items, 133 have one leaf, 54 have two, and only 13 have three or more. A range counts as two leaves, so leaf count measures the number of atomic comparisons, not nesting depth alone. This imbalance limits conclusions about complex composition. The appendix reports full metrics and data provenance.

5 Results

5.1 Typed Autoformalization and Selective Reliability

The neural selector has the highest overall logical accuracy: 154/200 (.770), at .980 coverage (Table 1). The rule parser retains 161/200 outputs, of which 135 are correct (.839). At the threshold fixed on development data, the neuro-symbolic selector retains 157/200 outputs. Of these, 140 are correct: .892 accepted accuracy (exact 95% CI [.832,.936]) at .785 coverage. Its accepted execution agreement is .964. It makes no accepted type or unit violations, as do the other systems in the table.

Comparison at the same coverage. The preceding operating points retain different numbers of items. We also compare the frozen rankings at exactly 156/200 items (.780 coverage). Neuro-symbolic, neural, and rule selection then return 140, 138, and 133 correct formulas: .897, .885,

System	Cov.	Overall logical	Accepted logical [exact 95% CI]	Accepted exec.	Viol.
Rule parser	.805	.675	.839 [.772, .892]	.929	.000
Neural selector	.980	.770	.786 [.722, .841]	.915	.000
Indexed parser	.415	.345	.831 [.733, .905]	.940	.000
Neuro-symbolic selector	.785	.700	.892 [.832, .936]	.964	.000

Table 1: Human-gold comparison ($n = 200$). The final threshold is selected on development data; brackets are exact two-sided binomial intervals.

and .853 accepted accuracy. The neuro-symbolic–neural difference is .013, with trial-clustered 95% CI [−.033,.056]. This interval includes zero. The .780 point is a descriptive matched-budget comparison, separate from the development-selected .785 threshold.

Across the attainable curves, the neural selector also has a slightly lower normalized area under the risk–coverage curve (AURC): .047 versus .051 for neuro-symbolic selection (lower is better). These values average risk over each system's available coverage range, ending at .980 and .990 respectively. Figure 2 shows the full curves. Neither the matched-coverage comparison nor AURC establishes a predictive advantage for the combined system. The chosen threshold provides a useful precision–coverage trade-off, while the audit trail supplies a separate benefit.

Variation across training seeds. For the verifier-gated neural selector, independent LoRA runs with seeds 42–44 produce accepted logical accuracy $.797 \pm .017$, execution agreement $.917 \pm .006$, and coverage $.943 \pm .040$ (mean $\pm$ sample SD). Every run has zero accepted type/unit violations. This experiment measures the neural component's variability; the multi-source selector uses the primary seed. Appendix B.3 gives the per-seed results.

5.2 Scope Errors and Missing Predicates

One expert categorized all 47 logically non-equivalent unconstrained-LoRA outputs using non-exclusive labels. Scope errors occur in 38 cases (80.9%) and missing predicates in 33 (70.2%); comparator direction appears in seven (14.9%) and units in only two (4.3%). This taxonomy describes the unconstrained neural outputs. The structural breakdown below shows that adding verification does not resolve the complex cases either.

Greedy neural accuracy falls from .917 for one leaf to .500 for two and .000 for all 13 formulas with three or more leaves. All three items containing NOT also fail. Neural verifier selection accepts 12 of the 13 larger formulas, and none is correct. Passing verification therefore does not ensure correct composition, even within the closed ontology.

The search diagnostic gives the same warning. Oracle@4 asks whether any of four neural candidates is logically correct. It succeeds on 0/13 larger formulas; adding the rule candidate raises this to only 1/13. The tested composition augmentation does not resolve the failure. Relative to standard verifier selection (.775 overall logical), shallow augmentation yields .735 ($\Delta = -.040$, 95% CI $[-.085, .000]$) and additional four–five-leaf training yields .700 ($\Delta = -.035$ from shallow, $[-.070, -.005]$); all remain 0/13 on formulas with three or more leaves. Because this subset contains only 13 items, these results identify observed failures rather than a precise estimate of performance on complex language. A larger independently annotated set, balanced by leaf count, nesting, coordination, and negation, is needed to test compositional generalization.

5.3 Decoding Changes Scope and Negation Errors

The 100-pair counterfactual suite changes one mathematical-language decision at a time. On both backbones, prompt decoding handles scope edits but fails all ten negation edits; typed+ontology decoding reverses the pattern, recovering all negation edits but none of the ten scope edits (Table 4). Similar aggregate scores therefore hide different kinds of error. The same pattern appears with Falcon3-7B. These are controlled edit families with only ten examples each; they do not demonstrate generalization to unseen natural compositions.

5.4 What the Symbolic Layer Contributes

The clearest benefits are explicit checks and an audit trail. The verifier reports which types, units, source spans, and intervals it checked. A reviewer can inspect a rejected formula or trace an accepted leaf back to the text. This makes failures easier to investigate, even when semantic accuracy does not

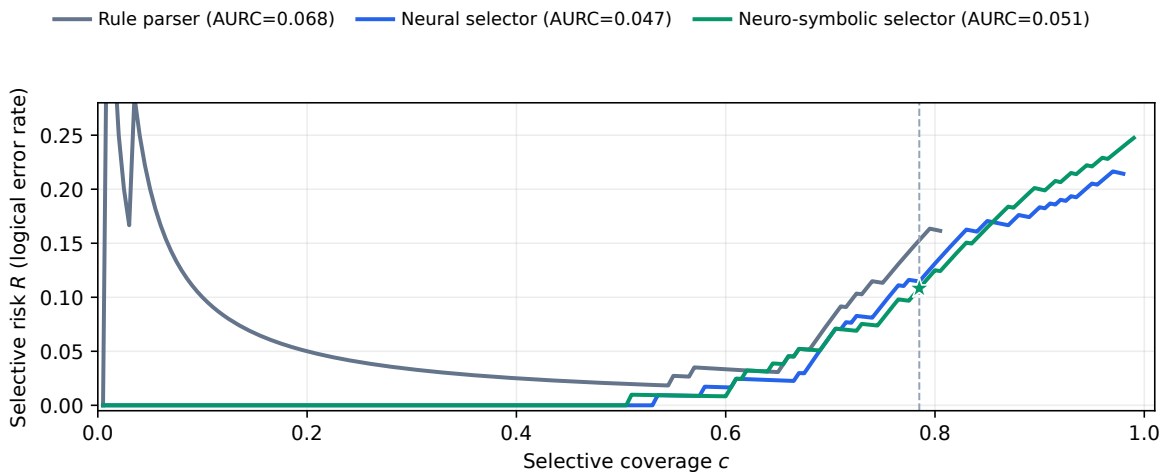


Figure 2: Full logical risk–coverage curves on human gold. Lower risk is better; the star is the development-selected neuro-symbolic operating point. Curves use frozen rankings, and coverage is the fraction of all 200 items retained.

Expert label	n	%	Expert label	n	%
Malformed or abstained	0	0.0	AND/OR/NOT scope	38	80.9
Variable grounding	2	4.3	Missing predicate	33	70.2
Comparator direction	7	14.9	Extra predicate	2	4.3
Numeric value	5	10.6	Evidence grounding	6	12.8
Unit	2	4.3			
Reviewed errors: 47			95 labels		

Gold structure	n	Rule	Neural	Neural+V cov./cond.
1 leaf	133	.865	.917	1.000/.947
2 leaves	54	.389	.500	.944/.549
3+ leaves	13	.077	.000	.923/.000
Contains AND	33	.242	.273	.939/.290
Contains OR	38	.395	.474	.947/.528
Contains NOT	3	.000	.000	1.000/.000

Table 2: Left: non-exclusive expert labels for all 47 neural logical errors. Right: overall exact logical equivalence by overlapping gold-structure groups; "contains" rows may overlap. Neural denotes greedy decoding; Neural+V denotes neural verifier selection and reports coverage/conditional correctness.

improve.

A separate matched-component study finds complementary rule and neural errors. Its evidence-linked hybrid reaches .760 overall logical accuracy, compared with .685 for its neural component and .670 for its rule component: gains of 7.5 points (95% CI [4.0,11.5]) and 9.0 points ([5.0,13.5]). These are different component configurations from Table 1. The result shows that evidence-linked selection can combine complementary candidates. Since it also changes the candidate pool, the gain cannot be attributed to verification alone. The record-integrity audit in the appendix separately tests whether the stored checks detect altered artifacts.

6 Related Work

Autoformalization and typed semantic parsing. Grammar-aware decoders generate programs through abstract syntax (Rabinovich et al., 2017); Neural Symbolic Machines combine neural induction and execution (Liang et al., 2017); and type constraints exclude ill-typed actions (Krishnamurthy et al., 2017). Work on typing variables in mathematical text (Stathopoulos et al., 2018) motivates our explicit mapping from lexical aliases to typed ontology symbols. We evaluate variable typing together with unit normalization and Boolean meaning, then examine the errors that remain when a formula is well typed.

Constrained decoding and execution guidance. Execution-guided decoding rejects programs that fail during execution (Wang et al., 2018); PICARD incrementally rejects inadmissible SQL tokens (Scholak et al., 2021); and grammar-constrained generation generalizes structured output constraints (Geng et al., 2023). We instantiate JSON-Schema masking with XGrammar (Dong et al., 2025). These methods guarantee membership in an encoded language, not correspondence between text and program. Our controlled edits show how a valid output can still express the wrong Boolean meaning.

Neuro-symbolic verification. Quan et al. (Quan et al., 2024) use a symbolic theorem prover to verify and refine LLM-generated explanations. We instead generate a typed formula under token masks and check the completed output before selection.

Criterion fragment	Human-gold structure	Neural failure
"ALT/AST $> 3\times$ upper limit"	$\mathrm{ALT} > 3 \lor \mathrm{AST} > 3$	Shared threshold is not distributed; AST leaf omitted.
"MI within 6 mo and LVEF $< 45\%$"	$\mathrm{MI}_{6m} \land \mathrm{LVEF} < 45$	First conjunct omitted.
"glucose 200–300 (A1C 7–10)"	Two ranges joined by AND (four leaves)	Parenthetical range and enclosing scope lost.
"bilirubin < 1.5; AST, ALT < 2.5"	Three conjunctive typed leaves	Coordination reduced to the first predicate.

Table 3: Representative adjudicated scope/omission errors. Slash coordination, ellipsis, parenthetical ranges, and shared comparators are characteristic scope phenomena in quantitative prose.

Backbone	Mode	Logical	Local edit	Negation	Scope
Llama-3.1-8B	Prompt	.900	.830	.000	1.000
Llama-3.1-8B	Typed+ontology	.900	.700	1.000	.000
Falcon3-7B	Prompt	.880	.780	.000	.900
Falcon3-7B	Typed+ontology	.870	.550	1.000	.000

Table 4: Frozen counterfactual evaluation ($n = 100$ per row; ten edits per family). Local edit requires that exactly the intended AST path change.

Our checks concern formal validity and source links. They do not prove that the text entails the formula. We evaluate this remaining semantic question against human gold.

Clinical quantitative language and selective prediction. Chia annotates trial criteria (Kury et al., 2020), Criteria2SQL maps criteria to queries (Yu et al., 2020), Valx structures numerical comparisons (Hao et al., 2016), and clinical NLP studies relations among numerical expressions (Dasgupta et al., 2020). We jointly evaluate typing, Boolean structure, exact execution, and evidence grounding. Coverage-conditioned reporting follows selective prediction (Geifman and El-Yaniv, 2019); we choose the operating point on development data and do not infer calibration from a hand-built score (Guo et al., 2017).

7 Conclusion

We present a typed neuro-symbolic pipeline that turns quantitative eligibility language into executable Boolean formulas. Its main contribution is a clear separation between neural interpretation, formal checks, and exact evaluation of meaning. Source-linked verification records make the output auditable, and selective prediction yields 89.2% logical correctness at 78.5% coverage.

The combined evidence from structural analysis, candidate search, augmentation, and controlled edits identifies Boolean scope and predicate omission as the main remaining bottlenecks. This diagnosis directs future work toward recovering complete predicate sets and their composition. Clinical eligibility language thus provides a controlled testbed for both executable semantic parsing and the study of mathematical-language understanding.

8 Limitations

Benchmark size and composition. The ontology contains 15 variables and 45 aliases. Of the 200 human-gold formulas, 187 have at most two leaves and only 13 have three or more; only three contain NOT. Aggregate results primarily describe shallow criteria. The neural system returns no correct formula on the 13 larger items; this small subset limits precise estimates across types of composition. The human packet re-annotates 199 held-out texts plus one replacement, rather than introducing an independently sampled corpus.

Verification and predictive accuracy. The verifier checks implemented formal constraints, lexical support, and record integrity. It can accept missing predicates or incorrect scope. Its interval reasoning is incomplete. Moreover, the neural selector already uses verifier checks, so comparing it with the combined selector does not isolate the effect of adding symbolic verification. The matched-coverage confidence interval includes zero, and the neural selector has better overall accuracy and AURC. The development precision criterion is not a guarantee of test precision.

Annotation and robustness. Two general-CS specialists annotated formal interpretations, followed by third-person adjudication; this is not clinical validation. Evidence boundaries vary between minimal predicates and enclosing clauses, although aligned spans overlap. One expert provided the error taxonomy. The main comparison uses training seed 42; the three-seed experiment measures the neural component, not the full multi-source selector. The second backbone and small controlled edit families provide limited replication, not broad model or language coverage.

Clinical scope. We use public eligibility text and no private patient records. The semantics assume complete records and the supported ontology. Clinical judgment, missing patient information, and criteria outside this fragment are beyond this study. The outputs are research artifacts, not clinical decisions.

References

Tirthankar Dasgupta, Ishani Mondal, Abir Naskar, and Lipika Dey. 2020. Exploring the role of semantic relationships in numerical expression recognition. In *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, pages 243–248. Association for Computational Linguistics.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ziyi Xu, Yilong Zhao, Ruihang Lai, and Tianqi Chen. 2025. XGrammar: Flexible and efficient structured generation engine for large language models. In *Proceedings of Machine Learning and Systems*, volume 7.

Yonatan Geifman and Ran El-Yaniv. 2019. SelectiveNet: A deep neural network with an integrated reject option. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2151–2159. PMLR.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10932–10952. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, 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. PMLR.

Tianyong Hao, Hongfang Liu, and Chunhua Weng. 2016. Valx: A system for extracting and structuring numeric lab test comparison statements from text. *Methods of Information in Medicine*, 55(3):266–275.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526. Association for Computational Linguistics.

Fabricio Kury, Alex Butler, Chi Yuan, Li-Heng Fu, Yingcheng Sun, Hao Liu, Ida Sim, Simona Carini, and Chunhua Weng. 2020. Chia, a large annotated corpus of clinical trial eligibility criteria. *Scientific Data*, 7(1):281.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33. Association for Computational Linguistics.

Xin Quan, Marco Valentino, Louise A. Dennis, and Andre Freitas. 2024. Verification and refinement of natural language explanations through LLM-symbolic theorem proving. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2933–2958. Association for Computational Linguistics.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149. Association for Computational Linguistics.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901. Association for Computational Linguistics.

Yiannos Stathopoulos, Simon Baker, Marek Rei, and Simone Teufel. 2018. Variable typing: Assigning meaning to variables in mathematical text. In *Proceedings of NAACL-HLT 2018, Volume 1 (Long Papers)*, pages 303–312. Association for Computational Linguistics.

Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. 2018. Robust text-to-SQL generation with execution-guided decoding. *arXiv preprint arXiv:1807.03100*.

Xiaojing Yu, Tianlong Chen, Zhengjie Yu, Huiyu Li, Yang Yang, Xiaoqian Jiang, and Anxiao Jiang. 2020. Dataset and enhanced model for eligibility criteria-to-SQL semantic parsing. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5829–5837, Marseille, France. European Language Resources Association.

A Evaluation Provenance and Audit Trail

This appendix follows one evaluation item from its source corpus to an adjudicated formula, a finite denotational test, and an auditable verification record. It first fixes data identity and counting conventions, then reports the label and artifact audits that support the main results.

Measure on 200 shared texts	Agreement
Predicate F1	.902
Boolean structure exact	.800
Semantic AST exact	.775
Logical equivalence	.785 [.730, .840]
Execution agreement	.922

Table 5: Source-derived labels compared with adjudicated human gold.

A.1 From Source Corpora to Human Gold

Chia is grouped by NCT trial, Criteria2SQL preserves its official assignments, and normalized duplicate texts connect source groups. The release gate requires zero normalized-text overlap, zero Chia NCT overlap, and no official test or development leakage. One contradictory duplicate is excluded in full.

The human benchmark contains 200 formulas with two independent annotations and adjudication. Source-derived labels are deterministic transformations used only for the secondary factorial; counterfactual labels are programmatically generated controlled edits. Table 5 quantifies how the source-derived formulas agree with human gold on the same packet texts, rather than treating the two label sources as interchangeable.

A.2 Counting and Denotational Evaluation

Ranges are expanded into two conjunctive leaves before leaf counts are computed. The size strata 1/2/3+ are disjoint and total 200. "Contains AND", "Contains OR", and "Contains NOT" inspect the complete compiled AST and overlap: a nested formula can contribute to several operator rows, and NOT may wrap a one-leaf formula. Operator counts therefore need not equal the number of non-atomic formulas.

For each item, one evaluation universe is fixed from the gold AST and every candidate in the rule, neural-$k = 4$, and indexed-$k = 2$ pools. Threshold cells are enumerated exhaustively; the largest Cartesian product contains 243 records. Logical equivalence is unchanged by this shared-universe rescore. All bootstrap intervals use 10,000 resamples with NCT-level Chia clusters and item-level Criteria2SQL clusters; exact binomial intervals are labelled separately. Table 6 gives the complete central metric vector.

A.3 Evidence and Verification-Record Integrity

Every record stores the ontology version, SHA-256 hashes of the criterion and canonical AST, and named checks with AST paths and offsets. Across 279 aligned annotator span pairs, all overlap: 59 are exact and 220 nested (character IoU .526; token F1 .659). Thus boundary disagreement reflects minimal-predicate versus enclosing-clause annotation rather than disjoint textual evidence.

The artifact audit injects 2,035 deterministic mutations spanning 11 implemented fault classes. Re-audit detects and localizes every injected fault, while 185 unmodified accepted records produce no false rejection. This experiment tests the verification guarantee in Section 3—hashes, paths, offsets, types, and encoded consistency checks—separately from semantic entailment.

A.4 Data Statement

Chia consists of public trial eligibility criteria, contains no patient records, and is distributed under CC BY 4.0 (Figshare DOI 10.6084/m9.figshare.11855817.v2). The Criteria2SQL repository carries Apache 2.0 for its code, but its README does not state a separate license for corpus text. The code repository provides download and transformation scripts. Users obtain the underlying corpus texts from their original repositories; downloaded texts and derived annotations are kept outside the public code repository. The artifacts are intended for research on formalization, not clinical decision making.

B System Reproduction and Robustness

The second part moves from evaluation artifacts to the systems that produce predictions. We specify the two auxiliary parsers and selector first, then give training details, independent-seed stability, and the mechanism factorial.

B.1 Rule, Indexed, and Selection Components

The rule parser contains five comparator families, three range templates, ten comparator-specific word-order patterns, and one negated-history pattern. It has no fitted parameters and was fixed before final adjudication. The indexed parser generates a depth-four Boolean skeleton with source character offsets, validates contiguous ordered slots, then fills one typed leaf per slot. Its final evidence spans are overwritten by the deterministic skeleton offsets.

The selector has 23 features: three source indicators; candidate index; verifier acceptance and reliability; evidence rate; mean log-probability; full-pool and available-pool support; agreement with the rule and another neural candidate; leaf count; depth; four root indicators; check count; pass and warning fractions; and mean/minimum check confidence. We standardize these features and fit ℓ_2-logistic regression in five development folds that keep each Chia NCT trial intact. The selected $C = .1$, sigmoid score, and threshold .568 are frozen before test scoring.

B.2 Training and Inference Configuration

Table 7 lists the training and inference settings.

B.3 Independent-Seed Stability

We independently train the same LoRA configuration with seeds 42, 43, and 44. Candidate generation uses seed 2026 in every run, isolating variability from parameter training. Table 8 evaluates the frozen verifier-gated neural selector on the same 200 human-gold items. Accepted logical and execution quality remain stable; most between-seed variation appears in coverage and consequently in overall logical accuracy.

B.4 Mechanism Factorial

The final diagnostic separates adaptation, token masks, and post-generation verification on 207 source-derived items. Typed decoding guarantees construction-time type safety, while adaptation supplies the semantic improvement over the base model. These cached generations use the same exhaustive threshold-cell evaluator as the human-gold experiment.

The appendices document the data, annotations, evaluation, and implementation needed to interpret the reported results.

System	Cov.	Pred. F1	Bool.	AST	Overall log.	Accepted log.	Sat.
Rule parser	.805	.756	.655	.650	.675	.839	.805
Neural selection	.980	.866	.830	.750	.770	.786	.980
Selective unified	.785	.732	.720	.690	.700	.892	.785

Table 6: Complete central metrics on adjudicated human gold.

Setting	Value
Primary model	Llama-3.1-8B-Instruct
Training data	3,698 records: 734 text-to-AST and 2,964 verifier-derived contrastive records
Development data	1,445 records: 254 text-to-AST and 1,191 contrastive records
Optimization	2 epochs; batch 2; gradient accumulation 4; learning rate 2×10^{-4}; linear schedule; no warmup; weight decay 0; BF16
LoRA	Rank 16; $\alpha = 32$; dropout .05; `q/k/v/o_proj`; 13,631,488 trainable parameters
Lengths	1,536 training tokens; 768 generated tokens; stop after one complete JSON object
Typed decoding	XGrammar 0.1.29; strict JSON-Schema masks; deterministic decoding
Ontology	Version 1.0.0; 15 variables; 45 aliases
Software	Transformers 4.47.0; PEFT 0.18.1; PyTorch 2.5.1

Table 7: Training and inference configuration.

Training seed	Cov.	Overall logical	Accepted logical	Accepted exec.	Viol.
42	.980	.770	.786	.914	.000
43	.950	.775	.816	.924	.000
44	.900	.710	.789	.914	.000
Mean $\pm$ SD	.943 $\pm$ .040	.752 $\pm$ .036	.797 $\pm$ .017	.917 $\pm$ .006	.000

Table 8: Independent LoRA training seeds on human gold ($n = 200$ per seed). The final row reports mean $\pm$ sample standard deviation.

System	Coverage	Logical equivalence	AST	Execution	Viol.
Base unconstrained	.000	.000	.000	.000	.000
Base unconstrained + verifier	.000	.000	.000	.000	.000
Base typed	1.000	.507	.464	.795	.000
Base typed + verifier	.556	.765	.748	.900	.000
LoRA unconstrained	1.000	.802	.778	.919	.019
LoRA unconstrained + verifier	.918	.847	.832	.942	.000
LoRA typed	.995	.738	.718	.908	.000
LoRA typed + verifier	.923	.775	.759	.916	.000

Table 9: Source-derived factorial ($n = 207$); metrics condition on coverage.

Which Mathematical Context Supports Target-Aligned Conjecture Generation? A Paired Ablation Study

Austin LaHue[1], Michael Burr[1], Luis David Garcia Puente[2], Vinita Gangaram Jansari[1], Benjamin Nye[2], Carlos Toxtli Hernandez[1]
[1]Clemson University, [2]Colorado College
jlahue@clemson.edu

Abstract

Generating mathematical conjectures from natural-language context is challenging because theorem-relevant information is distributed across multiple forms of mathematical context, including definitions, objects, assumptions, and constructions. We study which of these contextual signals help a model recover the specific claim supported by the surrounding setup, using 61 main theorems from 12 papers across four mathematical domains. For each theorem, we redact the result and represent the remaining setup using six context components. A paired ablation design compares full context, six leave-one-out conditions, and six single-component conditions, yielding 793 generated conjectures. Context components were largely complementary: removing one component usually had little effect, whereas retaining only one substantially reduced expected-theorem alignment. Local Mathematical Objects provided the strongest standalone signal, while Structural Constraints and Conditions produced the largest alignment decrease when removed. These findings suggest that target-aligned conjecture generation depends on combining information that identifies relevant objects with information that constrains admissible claims.

1 Introduction

Mathematical conjectures are formed within a local discourse of definitions, objects, assumptions, examples, and established results. Automated conjecturing has traditionally searched structured symbolic, programmatic, or formally verified spaces (Raayoni et al., 2021; Romera-Paredes et al., 2024; Lin et al., 2024), while recent mathematical language systems show that supplied definitions, theorems, and repository context can support proof generation and formal reasoning (Welleck et al., 2022; Hu et al., 2025). Less is known about which parts of natural-language mathematical context help a model formulate the particular claim that a paper goes on to prove.

We study this question through *target-aligned conjecture generation*. Given a redacted representation of the setup preceding a theorem, a model generates a formal conjecture without access to the theorem statement or proof. The task is narrower than open-ended mathematical discovery: success concerns whether the supplied context supports a mathematically substantive statement aligned with an existing hidden theorem. This controlled setting separates the ability to produce a plausible conjecture from the ability to recover the specific claim supported by the source context.

We conducted a paired ablation study over 61 main theorems from 12 papers spanning Commutative Algebra, Algebraic Geometry, Combinatorics, and Statistics Theory[1]. The redacted context is organized into six components and evaluated under full-context, leave-one-out, and single-component conditions. Results indicate that these components are largely complementary: removing one component generally had little effect, whereas retaining one alone substantially reduced alignment. Local Mathematical Objects supplied the strongest standalone signal, while removing Structural Constraints produced the largest alignment loss, suggesting distinct roles in identifying what a conjecture concerns and disambiguating what can be claimed. Together, these findings provide a component-level account of how theorem-local natural language supports target-aligned conjecture generation.

2 Related Work

2.1 Automated Conjecturing

Prior work on automated conjecturing includes heuristic discovery over graph invariants, theory exploration, and search within restricted formula spaces (Fajtlowicz, 1988; Larson and Van Cleemput, 2016; Colton et al., 2000; McCasland et al.,

[1]Supplementary materials are available at https://github.com/alahue/conjecture-ablation.

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 136–151
October 29, 2026 ©2026 Association for Computational Linguistics

2017; Raayoni et al., 2021). More recent systems combine learned guidance with executable search or formal verification (Romera-Paredes et al., 2024; Lin et al., 2024; Dong and Ma, 2025). These approaches largely generate candidates in structured, programmatic, or formal spaces. We instead study conjecture generation from the mixed natural and symbolic language preceding theorems in research papers, asking which semantic components of that local discourse support target alignment.

2.2 Mathematical Context

Mathematical text interleaves prose, formulas, local notation, and document-level dependencies. Work on premise selection and identifier grounding shows that definitions, theorems, lemmas, propositions, formula structures, and co-references help identify the information relevant to a mathematical task (Ferreira and Freitas, 2020, 2021; Asakura and Miyao, 2024). Context-grounded proof systems use such local and repository-level information to support proof generation (Welleck et al., 2022; Yang et al., 2023; Hu et al., 2025), while autoformalization and semantic-alignment work connects informal mathematical language with formal statements (Wu et al., 2022; Lu et al., 2025). Our study differs by fixing a hidden theorem target and ablating semantically defined components of its preceding setup. Because long-context use and mathematical reasoning can vary with information position and surface form (Liu et al., 2024; Williamson et al., 2025), we interpret the observed effects as properties of the constructed prompts rather than of mathematical context in general.

2.3 Ablation and Leakage

Our design parallels rationale-faithfulness evaluation: removing evidence measures its conditional contribution, whereas retaining it alone measures sufficiency (DeYoung et al., 2020). Accordingly, leave-one-out conditions estimate a component's contribution within rich context, and single-component conditions estimate standalone informativeness; the rankings may differ when information is complementary or redundant. Ablations can also create out-of-distribution inputs or preserve cues that directly reveal the target (Hase et al., 2020; Hsia et al., 2024). We therefore avoid causal interpretations and separately audit redacted seeds and model-generated generalization axes for theorem-shaped leakage.

2.4 Evaluation and Contamination

Generated conjectures must be assessed both for correspondence to the source theorem and for mathematical quality. Rubric-guided LLM judges can approximate human evaluation (Liu et al., 2023; Zheng et al., 2023), but scores remain sensitive to judge choice, repeated runs, and irrelevant persuasive language (Haldar and Hockenmaier, 2025; Hwang et al., 2025). We report repeated-judge, cross-model, and expert agreement and treat scores as prompt- and judge-dependent estimates rather than verified correctness. Finally, reconstruction may reflect pretraining exposure. Contamination work distinguishes exposure from exploitation and advocates perturbation-based controls (Magar and Schwartz, 2022; Qian et al., 2024; Gulati et al., 2025). Our publication-date holdout is exploratory, not proof of training-set inclusion or exclusion.

3 Method

3.1 Paper and Theorem Selection

We sampled 12 arXiv papers from four mathematical subject domains, with three papers selected from each:

- `math.AC` (Commutative Algebra) (Grimpen et al., 2026; Clarke and Kosta, 2024; Friedman et al., 2025);
- `math.AG` (Algebraic Geometry) (Jeronimo et al., 2024; Draisma et al., 2013; Salizzoni et al., 2025);
- `math.CO` (Combinatorics) (Sarkar et al., 2016; Le and Römer, 2023; Hollering et al., 2024);
- `math.ST` (Statistics Theory) (Brysiewicz and Kim, 2026; Alexandr et al., 2022; Duarte and Solus, 2021).

Within each domain, papers were additionally required to have at least one Mathematics Subject Classification (MSC) code in one of the following families: 13Pxx (Computational aspects and applications of commutative rings), 14Qxx (Computational aspects in algebraic geometry), 68Wxx (Algorithms in computer science), or 62Rxx (Statistics on algebraic and topological structures). Among the selected papers classified within 62Rxx, all were assigned to 62R01 (Algebraic statistics).

For each paper, the theorem-selection stage of our pipeline classified all theorem-like environments and identified the paper's *main proved theorems*. A result qualified if it was presented as a central contribution of the paper, proved within

the paper, and not merely a quoted result, supporting lemma or proposition, corollary, example, or background theorem. This process identified 61 theorem targets across the 12 papers. Individual papers contributed between 3 and 7 targets. The domain totals were 14 for `math.ST`, 17 for `math.AC`, 16 for `math.AG`, and 14 for `math.CO`.

Four of the selected papers, comprising 21 theorem targets, were posted after May 2025, a knowledge-cutoff date reported for DeepSeek-V4-Pro by third-party model documentation (Ng, 2026). We treat these papers as a temporal-holdout cohort and compare them with the 8 earlier papers.

A human reviewer checked all 61 selected theorem targets against the source papers and confirmed that each was an eligible main proved theorem. We did not assess whether the pipeline identified every eligible theorem in the selected papers. One statement was incompletely extracted, while minor transcription issues included lost LaTeX formatting in two papers and added definitions in two papers.

3.2 Conjecture-Generation Pipeline

Figure 1 summarizes the end-to-end pipeline. We used Codex (OpenAI, 2025b) as the orchestration harness, configured to call DeepSeek-V4-Pro (DeepSeek-AI, 2026) through a vLLM-served endpoint (Kwon et al., 2023). DeepSeek-V4-Pro performed theorem identification, object-map construction, redaction, and conjecture generation under each experimental condition. The generated conjecture and corresponding source theorem were then provided to DeepEval (Ip and Vongthongsri, 2026), with Gemma-4-31B (Gemma Team, 2026) serving as the judge, to produce theorem-conditioned conjecture-quality and expected-theorem-alignment scores.

For each theorem target, the generation pipeline performed the following steps:

1. Construct a theorem-specific object map containing the source theorem, relevant definitions and notation, mathematical objects, assumptions, supporting results and constructions, and proof dependencies.

2. Convert the object map into a redacted seed by removing information that could reveal the target theorem. The redacted content included the theorem statement and conclusion, theorem identifiers, proof strategy, dependency chain, final classifications, and exact result-specific formulas or equivalences.

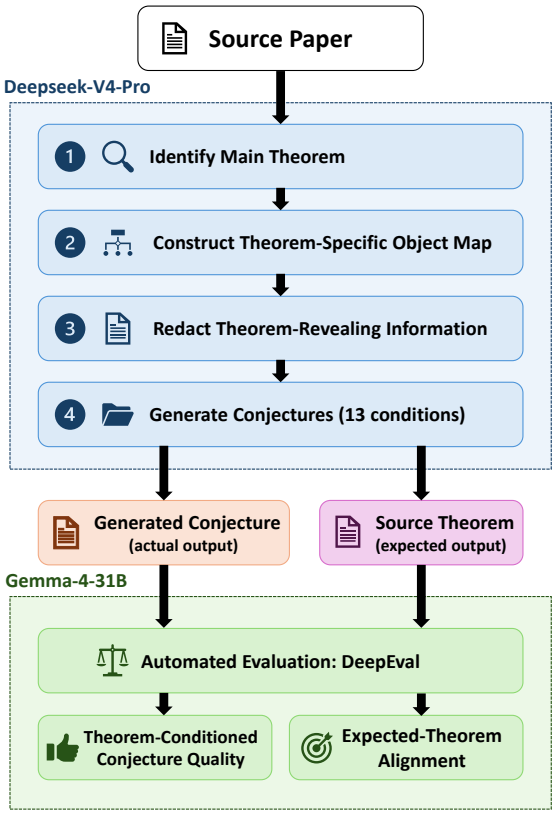


Figure 1: Overview of the conjecture-generation and evaluation pipeline.

3. Organize the remaining setup information into six mathematical-context components.

4. Generate one conjecture from the selected components for each experimental condition.

The conjecture-generation model was provided only with the redacted seed for the corresponding condition. The original paper, theorem map, unredacted object map, source theorem, proof strategy, and dependency chain were not included in the model input, and web search was disabled. It generated exactly one formal conjecture per theorem and condition. The substantive classification, redaction, and generation instructions are summarized in Appendix E.

Table 1 shows a source theorem identified by the pipeline and the conjecture generated from its redacted mathematical-context components.

3.3 Mathematical-Context Components

Each redacted seed represented the theorem's preceding context using six components:

1. **Ambient Setting**: the broad mathematical setting and background assumptions, without

identifying the target theorem;

2. **Core Definitions and Notation**: definitions and notation required to understand the local problem;
3. **Local Mathematical Objects**: structures, quantities, maps, operators, relations, functions, families, or other mathematical entities that define the local problem setting;
4. **Structural Constraints and Conditions**: hypotheses, restrictions, and validity conditions defining the local problem setting;
5. **Auxiliary Objects and Constructions**: supporting examples, intermediate objects, derived constructions, or families used to study, illustrate, or relate the local mathematical objects;
6. **Potential Generalization Axes**: model-generated directions for forming conjectures, such as varying parameters, weakening assumptions, changing the ambient object, or examining exceptional cases.

The first five components were summarized from the theorem-specific source material after redaction. The Potential Generalization Axes were synthesized by DeepSeek-V4-Pro from the redacted mathematical setup.

All 61 redacted theorem seeds underwent a leakage audit (Appendix A). Fifty-eight seeds (95.1%) were flagged for potential inferential leakage and reviewed, but all were retained after mitigation; no audit summary reported an explicit hidden theorem statement or final conclusion remaining in a seed.

Source theorem

Let $U \subseteq V \subseteq R^d$ be submodules and $\prec$ a fixed monomial order on R^d. Then V has a unique reduced Gröbner basis relative to U.

Generated conjecture

Let $\Bbbk$ be a field, let $R = \Bbbk[X_1, \ldots, X_n]$, and let $U \subseteq V \subseteq R^d$ be nested submodules of the free module R^d. Fix a monomial order $\prec$ on $\mathrm{Mon}(R^d)$. Then there exists a reduced relative Gröbner basis H of V with respect to U.
Moreover, this reduced relative Gröbner basis is unique: if H and H' are two reduced relative Gröbner bases of V relative to U, with respect to the same monomial order $\prec$, then

$$H = H'.$$

Table 1: Example of a source theorem (Grimpen et al., 2026) and the corresponding generated conjecture.

Because Potential Generalization Axes were synthesized by the same model used for conjecture generation, we audited this component separately for theorem-shaped information (Appendix B). Of the 61 seeds, 12 (19.7%) were classified as direct theorem-shaped, 24 (39.3%) as strongly directional, and 25 (41.0%) as low or contextual. Accordingly, results involving this component may reflect theorem-directed cues in addition to genuine support for conjecture generation and are therefore interpreted cautiously.

3.4 Ablation Design

For each of the 61 theorem targets, we generated conjectures under 13 context conditions:

- one **full-context** condition containing all six components;
- six **leave-one-out** conditions, each excluding one component while retaining the other five;
- six **single-component** conditions, each retaining one component while excluding the other five.

This yielded $61 \times 13 = 793$ generated conjectures. The design is paired: all 13 conditions for a theorem were derived from the same theorem-specific object map and evaluated against the same source theorem. Leave-one-out conditions estimate a component's contribution in the presence of otherwise rich context, whereas single-component conditions estimate its standalone informativeness.

Because redaction cannot prevent recovery of a theorem from model pretraining, we conducted an exploratory temporal-holdout analysis. We compared full-context performance and the mean effects of the leave-one-out and single-component conditions between the four post-cutoff papers and the eight earlier papers. Cohort comparisons used paper-level summaries so that each paper received equal weight. Because DeepSeek has not published an official knowledge cutoff, publication date does not establish actual training-set inclusion, and the cohorts are not balanced by domain, we treat this analysis as exploratory.

3.5 Automated Conjecture Evaluation

We evaluated conjectures using DeepEval with Gemma-4-31B as the primary judge and a temperature of zero, following exploratory comparisons among candidate judges. DeepEval received

the generated statement as the *actual output* and the original source theorem as the *expected output*. Scores were assigned on a scale from 0 to 1 in increments of 0.1, with higher values indicating stronger performance.

We used two metrics:

- **Expected-Theorem Alignment**: implemented using DeepEval's documented correctness criterion: "Determine whether the actual output is factually correct based on the expected output." (Confident AI, 2023) Because this measure compares a generated conjecture with a target theorem rather than independently verifying the conjecture, we interpret it as LLM-estimated theorem-alignment rather than verified mathematical correctness.

- **Theorem-Conditioned Conjecture Quality**: a custom, target-aware rubric assessing whether the output contains a clear and substantive mathematical claim, its simple-case plausibility, precision, potential usefulness, and relevance to the source theorem. Unlike expected-theorem alignment, this metric can reward a meaningful specialization, close variant, or plausible generalization even when it does not accurately recover the target result. Because the expected theorem is available to the judge, this metric does not measure source-independent novelty or intrinsic conjecture quality. The complete rubric is provided in Appendix D.

Both metrics are conditioned on the expected theorem, but operationalize different outcomes. Expected-theorem alignment is reconstruction-oriented, whereas theorem-conditioned conjecture quality permits mathematically meaningful departures from the target theorem.

3.6 Expert Validation

We evaluated agreement between the automated theorem-conditioned quality judge and an expert mathematician using an independent validation set of 15 conjectures. These conjectures were generated separately using o3-mini (OpenAI, 2025c) and were not sampled from the ablation-study outputs or assigned to experimental conditions. The expert applied the same theorem-conditioned quality rubric used by the automated judge and was blinded to the LLM-generated scores.

Expert and Gemma theorem-conditioned quality scores were moderately positively correlated, $r(13) = .60$, $p = .017$, 95% CI $[.13, .85]$. Absolute agreement was also moderate using a two-way mixed-effects, absolute-agreement, single-measure intraclass correlation, $\mathrm{ICC}(A, 1) = .54$, with a bootstrap 95% CI of $[.09, .77]$. The mean absolute error was 0.27, and Gemma assigned scores that were, on average, 0.16 points higher than the expert. We therefore interpret this validation as preliminary evidence of evaluator agreement rather than definitive validation, given the single expert, small sample, and use of an independently generated validation set.

The expert did not evaluate expected-theorem alignment because doing so would require substantial proof development or counterexample search. The Expert theorem-conditioned quality scores were positively associated with the automated expected-theorem-alignment scores, $r(13) = .58$, $p = .022$, 95% CI $[.10, .84]$; however, because these variables measure different constructs, we treat this as an exploratory association rather than direct validation of the alignment metric.

3.7 Judge Reliability and Cross-Model Agreement

To assess evaluator reliability, we rescored the full-context and six leave-one-out outputs from four papers, with one paper selected from each domain. We first repeated the evaluation using Gemma-4-31B with the same prompts and settings. We then evaluated the same outputs using GPT-5-mini (OpenAI, 2025a) as an alternative judge. For each metric, we assessed score association using Pearson's r and absolute agreement using $\mathrm{ICC}(A, 1)$.

We also piloted GLM-5.1-FP8 (Z.ai, 2026; GLM-5 Team, 2026) as an expected-theorem-alignment judge. Its lower agreement with Gemma appeared to reflect a stricter treatment of claims absent from the expected theorem. We therefore report it as an exploratory diagnostic of judge and prompt sensitivity (Appendix C).

3.8 Statistical Analysis

Because each theorem target was evaluated under every context condition, analyses were paired at the theorem level. For leave-one-out conditions, we report mean differences from the full-context baseline, paired effect sizes (d_z), and 95% confidence intervals for the paired differences. Overall differences among the full-context and six leave-one-out

conditions were assessed for each evaluation metric using Friedman tests, followed by Wilcoxon signed-rank comparisons with the full-context baseline and Holm correction across the six comparisons.

Because theorem targets from the same paper share paper-level context, we treated them as clustered rather than independent observations. To prevent papers with more targets from receiving greater weight, we repeated the primary analyses using paper-level mean scores, with each paper weighted equally. Domain comparisons were computed from these summaries and treated as exploratory because each domain contained only three papers. For each metric, differences in component-removal effects across domains were assessed using Kruskal–Wallis tests with Holm correction across the six components. Single-component conditions were analyzed descriptively to estimate standalone informativeness.

4 Results

4.1 Context Components Are Largely Complementary

Under the full-context condition, mean theorem-conditioned conjecture quality was 0.931, and mean expected-theorem alignment was 0.607. Figure 2 reports paired theorem-level differences from these baselines. Complete condition-level results are reported in Appendix F.

Leave-one-out ablations produced only small changes in either metric. The omnibus tests were not significant for theorem-conditioned quality, $\chi^2(6) = 5.28$, $p = .509$, or expected-theorem alignment, $\chi^2(6) = 9.75$, $p = .135$. Removing Structural Constraints and Conditions produced the largest alignment decrease, $\Delta = -0.087$, 95% CI $[-0.154, -0.020]$, but the comparison did not remain significant after Holm correction, $p_{\text{Holm}} = .055$. No other leave-one-out effect was clearly distinguishable from zero.

Equal-weight paper-level analyses yielded the same qualitative pattern. The omnibus tests remained nonsignificant, and removing Structural Constraints again produced the largest alignment decrease, $\Delta = -0.085$, $p_{\text{Holm}} = .105$.

The single-component conditions revealed a distinction between standalone and complementary information. Local Mathematical Objects provided the strongest standalone signal, with an alignment mean of 0.457. Structural Constraints were less informative alone, with an alignment mean of 0.348, despite producing the largest decrease when removed from full context. Thus, Local Mathematical Objects supplied the most directly usable information, whereas Structural Constraints contributed by disambiguating otherwise rich context.

4.2 Component Importance Varies by Domain

Domain-level analyses were exploratory because each domain contained only three papers. Full-context performance was broadly similar across domains, but the component associated with the largest alignment loss varied (Table 2). Structural Constraints were most consequential in the two algebraic domains, whereas Definitions and Notation were most consequential in `math.CO`, and Local Mathematical Objects were most consequential in `math.ST`. The complete domain-level results are reported in Appendix G.

Potential Generalization Axes showed the clearest qualitative domain difference. Removing them reduced alignment in `math.AC` and `math.AG`, but increased alignment in `math.CO` and `math.ST`. The removal effect did not differ significantly across domains after Holm correction, $p_{\text{Holm}} = .553$.

4.3 Held-Out Papers Show Suggestive Sensitivity to Context Reduction

Full-context performance was similar for the post-cutoff and earlier-paper cohorts: expected-theorem alignment was 0.630 and 0.619, respectively, while theorem-conditioned quality was 0.929 and 0.942. However, the post-cutoff papers were more sensitive to incomplete context. Their mean leave-one-out alignment effect was -0.083, compared with approximately zero for the earlier papers, and their mean single-component effect was -0.346, compared with -0.218. Theorem-conditioned quality effects were nearly identical between cohorts. The paper-level cohort differences were not significant for either the leave-one-out ($p = .214$) or single-component ($p = .087$) analyses. These descriptive patterns are consistent with greater reliance on the

Domain	Full	Largest loss	Δ	Best standalone
`math.AC`	.611	Constraints	-.162	Local objects
`math.AG`	.571	Constraints	-.121	Definitions
`math.CO`	.644	Definitions	-.086	Constraints
`math.ST`	.664	Local objects	-.096	Definitions

Table 2: Exploratory domain-level expected-theorem alignment. Full is the full-context baseline. Each domain contains three papers.

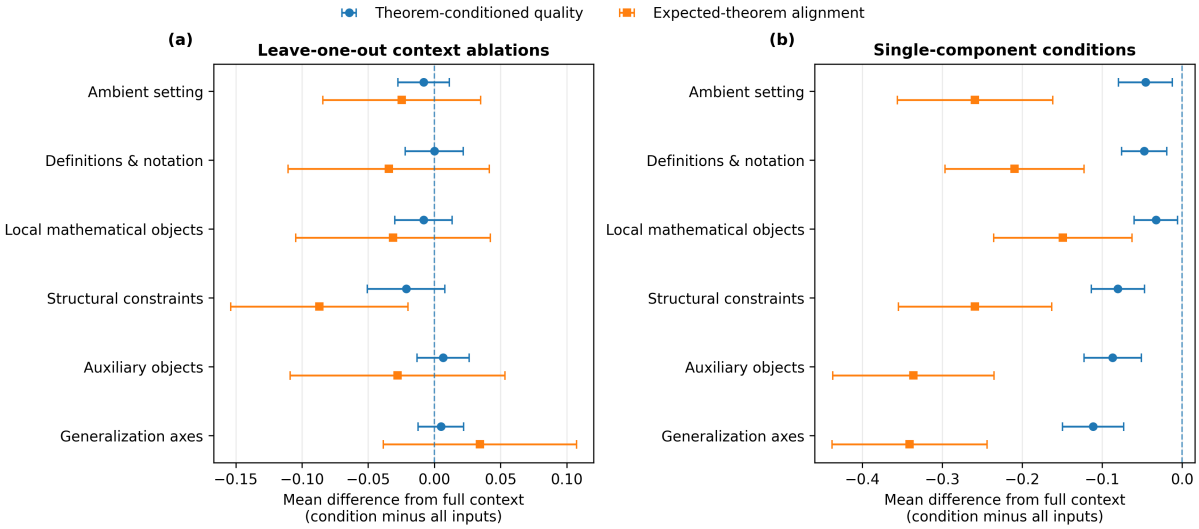


Figure 2: Paired theorem-level differences from the full-context condition with 95% confidence intervals. Panel (a) removes one context component, whereas panel (b) retains only the named component. Negative values indicate lower scores than under full context.

supplied context for temporally held-out papers, but do not establish whether any individual paper was present in the model's training data.

4.4 Judge Agreement Varies by Metric

Repeated Gemma evaluation showed high reliability for both metrics (Table 3). Cross-model agreement with GPT-5-mini was lower: alignment scores showed moderate absolute agreement, whereas absolute quality scores were sensitive to judge choice.

5 Discussion

This study examined which forms of the theorem-local mathematical context support conjectures aligned with a hidden target theorem. Across 61 theorem targets and 13 paired conditions, the results indicate that context components are largely complementary rather than individually indispensable. Removing one component from an otherwise complete context generally produced only small changes, whereas retaining a single component substantially reduced expected-theorem alignment.

Comparison	Metric	r	ICC
Gemma repeat	Alignment	.921	.921
	Quality	.840	.841
GPT-5-mini	Alignment	.784	.731
	Quality	.568	.212

Table 3: Repeated-judge reliability and cross-model agreement.

Thus, the small leave-one-out effects do not imply that context is unimportant. Instead, multiple components appear to provide overlapping information and partially compensate for one another.

The contrast between the two evaluation metrics further clarifies this result. Full context produced high theorem-conditioned conjecture quality but only moderate expected-theorem alignment. Even under single-component conditions, the model often generated plausible and relevant mathematical statements despite failing to recover the particular result proved in the source paper. Limited context may therefore be sufficient for producing a mathematically substantive continuation, while richer context is needed for target-aligned generation. These are related but distinct capabilities.

5.1 Objects Propose; Constraints Disambiguate

The clearest functional distinction emerged between Local Mathematical Objects and Structural Constraints and Conditions. Local Mathematical Objects provided the strongest standalone alignment signal, suggesting that theorem-relevant objects, maps, quantities, and families help define the broad mathematical subject and were often sufficient for the model to produce a related claim.

Structural Constraints played a different role. Although less informative in isolation, their removal from the full context produced the largest decrease in expected-theorem alignment. This pattern suggests that constraints are primarily disambiguating:

mathematical objects define a space of possible claims, while hypotheses, restrictions, and validity conditions help determine which relation among those objects is intended. Objects may therefore support candidate formation, whereas constraints help select among otherwise plausible candidates.

This distinction also shows why component importance cannot be represented by a single ranking. Standalone performance reflects how directly a component supports generation by itself, while leave-one-out effects reflect its contribution when combined with the remaining context. For target-aligned conjecture generation, direct informativeness and conditional contribution appear to be separate properties.

The exploratory domain results suggest that these functions may vary across mathematical fields. Structural Constraints produced the largest alignment losses in Commutative Algebra and Algebraic Geometry, while Definitions and Notation were most consequential in Combinatorics and Local Mathematical Objects in Statistics Theory. Because each domain contained only three papers, these patterns are not conclusive, but they motivate domain-adaptive approaches to selecting and organizing mathematical context.

The Potential Generalization Axes showed less consistent effects. Their removal reduced alignment in the algebraic domains but increased it in Combinatorics and Statistics Theory, and they were weak as a standalone component. Generalization directions may sometimes focus generation toward the target, but they may also broaden the candidate space. Because these axes were model-generated rather than directly extracted from the source, they are better treated as a conjecturing scaffold than as equivalent to the five source-derived components.

5.2 Context Dependence and Evaluation

Earlier and post-cutoff papers achieved similar full-context performance, but the post-cutoff cohort showed larger alignment losses when context was reduced. This pattern is consistent with the model relying more heavily on the supplied mathematical setup when the target material is less familiar from training. It does not establish whether any paper was included in the training data, and the cohort differences were not statistically conclusive. The temporal analysis therefore provides evidence about context sensitivity rather than contamination itself. Controlled perturbations, such as systematic symbol renaming or paraphrasing, would provide stronger tests of whether generation depends on supplied mathematical meaning, familiar lexical patterns, or prior exposure.

Evaluator agreement also differed by metric. Expected-theorem alignment showed stronger repeated and cross-model agreement than theorem-conditioned conjecture quality. Alignment asks a constrained question about correspondence to a supplied theorem, whereas quality is a broader judgment of clarity, simple-case plausibility, precision, potential usefulness, and relevance to the source theorem. Expected-theorem alignment therefore provides a more stable basis for comparing ablation conditions, while theorem-conditioned quality indicates whether low-alignment outputs remain mathematically substantive.

Neither metric independently establishes mathematical truth or novelty. Future evaluation could score the existing quality dimensions separately rather than combining them into one holistic judgment. It could also classify each output as equivalent to the target theorem, a specialization, a generalization, or a different but related result. Proof attempts, counterexample search, and source-blind expert review would further distinguish target recovery from mathematical validity.

Overall, the findings suggest that conjecture-generation systems may benefit from organizing context according to its mathematical role. Local Mathematical Objects provide an efficient basis for identifying what to conjecture about, while Structural Constraints, definitions, and ambient information narrow the admissible claims. Because component effects varied across domains, adaptive context assembly may be more effective than a fixed minimal prompt. Target alignment appears to emerge from the interaction between context that identifies relevant objects and context that constrains admissible claims.

6 Conclusion

We presented a paired ablation study of theorem-local context for target-aligned conjecture generation. Across 61 theorem targets, Local Mathematical Objects were most informative in isolation, while Structural Constraints contributed most within richer context. Future work should test whether these patterns generalize across additional models, domains, context decompositions, and source-blind evaluations of validity and novelty.

Limitations

Both evaluation metrics are conditioned on the expected theorem. The theorem-conditioned quality score therefore does not measure source-independent novelty, usefulness, or mathematical validity, while expected-theorem alignment remains sensitive to judge and prompt interpretation. This sensitivity was reflected in the stricter behavior of GLM-5.1-FP8 relative to Gemma-4-31B and GPT-5-mini. Expert validation was also limited to one mathematician and 15 conjectures generated separately by o3-mini, so agreement may not generalize to the ablation-study outputs. The expert did not independently validate expected-theorem alignment, as doing so would require proof development or counterexample search. Accordingly, the reported scores should be treated as target-conditioned estimates rather than definitive assessments of conjecture quality or correctness.

The ablations do not isolate the causal importance of each component. Removing or retaining components changes prompt length, coherence, redundancy, and resemblance to natural mathematical exposition. Single-component prompts may be especially unnatural, while leave-one-out prompts may allow overlapping information to compensate for the removed component. The study is also limited to 61 theorem targets from 12 papers, four computationally oriented domains, one generation model, and one conjecture per theorem–condition pair, so within-condition generation variability was not estimated. Domain comparisons are therefore exploratory, and the results may depend on the selected papers, prompts, and component-construction procedure. The LLM-based selection pipeline was validated only for the targets it selected; because recall was not assessed, it may have omitted other eligible main theorems.

The temporal-holdout analysis relies on a third-party knowledge-cutoff estimate because DeepSeek has not disclosed an official cutoff for DeepSeek-V4-Pro. The publication date cannot establish whether a paper or related material appeared in training, and the temporal cohorts also differ in domain composition. These results should therefore be interpreted as evidence of differing context sensitivity rather than direct evidence of training-data contamination.

Potential Generalization Axes present an additional leakage risk because they were synthesized from theorem-specific material by the same model used for generation. The audit classified 19.7% of seeds as directly theorem-shaped and 39.3% as strongly directional. Effects involving this component may therefore reflect theorem-directed cues as well as support for generalization. More broadly, the six components represent one operationalization of mathematical context; other reasonable decompositions may produce different results.

Acknowledgments

This work was supported by the National Science Foundation under Award No. 2432704. This research used in part resources on the Palmetto 2 cluster at Clemson University under National Science Foundation awards MRI 1228312, II NEW 1405767, MRI 1725573, and MRI 2018069. The views expressed in this article do not necessarily represent the views of NSF or the United States government.

The authors used ChatGPT to assist with refining the language and organization of the manuscript and to support discussion of statistical analyses and presentation of results. All generated suggestions were reviewed, verified, and revised by the authors, who take full responsibility for the content of the paper. Generative AI models used as part of the experimental methodology are described separately in Section 3.2.

References

Yulia Alexandr, Eliana Duarte, and Julian Vill. 2022. Decomposable context-specific models. *Preprint*, arXiv:2210.11521.

Takuto Asakura and Yusuke Miyao. 2024. What is needed for intra-document disambiguation of math identifiers? In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 17500–17512, Torino, Italia. ELRA and ICCL.

Taylor Brysiewicz and Juhee Kim. 2026. How to recover a permutation group amidst errors. *Preprint*, arXiv:2602.23505.

Oliver Clarke and Dimitra Kosta. 2024. Distance reducing markov bases. *Preprint*, arXiv:2406.17730.

Simon Colton, Alan Bundy, and Toby Walsh. 2000. On the notion of interestingness in automated mathematical discovery. *International Journal of Human-Computer Studies*, 53(3):351–375.

Confident AI. 2023. Introduction to LLM evaluation metrics. DeepEval documentation. Accessed: 2026-08-05.

DeepSeek-AI. 2026. DeepSeek-V4: Towards highly efficient million-token context intelligence. *Preprint*, arXiv:2606.19348.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.

Kefan Dong and Tengyu Ma. 2025. STP: Self-play LLM theorem provers with iterative conjecturing and proving. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 14114–14136. PMLR.

Jan Draisma, Emil Horobet, Giorgio Ottaviani, Bernd Sturmfels, and Rekha R. Thomas. 2013. The euclidean distance degree of an algebraic variety. *Preprint*, arXiv:1309.0049.

Eliana Duarte and Liam Solus. 2021. Representation of context-specific causal models with observational and interventional data. *Preprint*, arXiv:2101.09271.

Siemion Fajtlowicz. 1988. On conjectures of Graffiti. In *Graph Theory and Its Applications: East and West*, volume 38 of *Annals of Discrete Mathematics*, pages 113–118. Elsevier.

Deborah Ferreira and André Freitas. 2020. Premise selection in natural language mathematical texts. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7365–7374, Online. Association for Computational Linguistics.

Deborah Ferreira and André Freitas. 2021. STAR: Cross-modal statement representation for selecting relevant mathematical premises. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3234–3243, Online. Association for Computational Linguistics.

Hannah Friedman, Bernd Sturmfels, and Maximilian Wiesmann. 2025. Squared linear models. *Preprint*, arXiv:2505.19351.

Gemma Team. 2026. Gemma 4 technical report. *Preprint*, arXiv:2607.02770.

GLM-5 Team. 2026. GLM-5: From vibe coding to agentic engineering. *Preprint*, arXiv:2602.15763.

Fritz Grimpen, Matthias Orth, and Anastasios Stefanou. 2026. Relative Gröbner bases of modules and applications in persistence theory. *Preprint*, arXiv:2603.27149.

Aryan Gulati, Brando Miranda, Eric Chen, Emily Xia, Kai Fronsdal, Bruno De Moraes Dumont, and Sanmi Koyejo. 2025. Putnam-AXIOM: A functional and static benchmark for measuring higher level mathematical reasoning in LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 20723–20747. PMLR.

Rajarshi Haldar and Julia Hockenmaier. 2025. Rating roulette: Self-inconsistency in LLM-as-a-judge frameworks. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 24986–25004, Suzhou, China. Association for Computational Linguistics.

Peter Hase, Shiyue Zhang, Harry Xie, and Mohit Bansal. 2020. Leakage-adjusted simulatability: Can models generate non-trivial explanations of their behavior in natural language? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4351–4367, Online. Association for Computational Linguistics.

Benjamin Hollering, Joseph Johnson, and Liam Solus. 2024. Hyperplane representations of interventional characteristic imset polytopes. *Preprint*, arXiv:2404.18500.

Jennifer Hsia, Danish Pruthi, Aarti Singh, and Zachary C. Lipton. 2024. Goodhart's law applies to NLP's explanation benchmarks. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1322–1335, St. Julian's, Malta. Association for Computational Linguistics.

Jiewen Hu, Thomas Zhu, and Sean Welleck. 2025. miniCTX: Neural theorem proving with (long-)contexts. In *The Thirteenth International Conference on Learning Representations*.

Yerin Hwang, Dongryeol Lee, Taegwan Kang, Yongil Kim, and Kyomin Jung. 2025. Can you trick the grader? adversarial persuasion of LLM judges. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 14632–14651, Suzhou, China. Association for Computational Linguistics.

Jeffrey Ip and Kritin Vongthongsri. 2026. DeepEval: The open-source LLM evaluation framework. Software, version 4.1.5, released 2026-07-16.

Gabriela Jeronimo, Leonardo Lanciano, and Pablo Solernó. 2024. On the geometric degree of the tangent bundle of a smooth algebraic variety. *Preprint*, arXiv:2403.10661.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626. Association for Computing Machinery.

Craig E. Larson and Nicolas Van Cleemput. 2016. Automated conjecturing I: Fajtlowicz's dalmatian heuristic revisited. *Artificial Intelligence*, 231:17–38.

Dinh Van Le and Tim Römer. 2023. Equivariant lattice bases. *Preprint*, arXiv:2309.07246.

Xiaohan Lin, Qingxing Cao, Yinya Huang, Zhicheng Yang, Zhengying Liu, Zhenguo Li, and Xiaodan Liang. 2024. ATG: Benchmarking automated theorem generation for generative language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4465–4480, Mexico City, Mexico. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Jianqiao Lu, Yingjia Wan, Yinya Huang, Jing Xiong, Zhengying Liu, and Zhijiang Guo. 2025. FormalAlign: Automated alignment evaluation for autoformalization. In *The Thirteenth International Conference on Learning Representations*.

Inbal Magar and Roy Schwartz. 2022. Data contamination: From memorization to exploitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 157–165, Dublin, Ireland. Association for Computational Linguistics.

Roy L. McCasland, Alan Bundy, and Patrick F. Smith. 2017. MATHsAiD: Automated mathematical theory exploration. *Applied Intelligence*, 47(3):585–606.

Ann Ng. 2026. How to use DeepSeek V4 Pro from DeepSeek with API key on TypingMind. Accessed: 2026-08-03.

OpenAI. 2025a. GPT-5 system card. Accessed: 2026-08-05.

OpenAI. 2025b. Introducing Codex. Accessed: 2026-08-04.

OpenAI. 2025c. OpenAI o3-mini. Accessed: 2026-08-06.

Kun Qian, Shunji Wan, Claudia Tang, Youzhi Wang, Xuanming Zhang, Maximillian Chen, and Zhou Yu. 2024. VarBench: Robust language model benchmarking through dynamic variable perturbation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16131–16161, Miami, Florida, USA. Association for Computational Linguistics.

Gal Raayoni, Shahar Gottlieb, Yahel Manor, George Pisha, Yoav Harris, Uri Mendlovic, Doron Haviv, Yaron Hadad, and Ido Kaminer. 2021. Generating conjectures on fundamental constants with the Ramanujan Machine. *Nature*, 590:67–73.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. 2024. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475.

Flavio Salizzoni, Luca Sodomaco, and Julian Weigert. 2025. Nonlinear Kalman varieties. *Preprint*, arXiv:2512.16540.

Kaushik Sarkar, Charles J. Colbourn, Annalisa De Bonis, and Ugo Vaccaro. 2016. Partial covering arrays: Algorithms and asymptotics. *Preprint*, arXiv:1605.02131.

Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. 2022. NaturalProver: Grounded mathematical proof generation with language models. In *Advances in Neural Information Processing Systems*, volume 35.

Dane A. Williamson, Yangfeng Ji, and Matthew B. Dwyer. 2025. Syntactic blind spots: How misalignment leads to LLMs' mathematical errors. In *Proceedings of the 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025)*, pages 1–14, Suzhou, China. Association for Computational Linguistics.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*, volume 35.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J. Prenger, and Animashree Anandkumar. 2023. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, volume 36.

Z.ai. 2026. GLM-5.1-FP8 model card. Hugging Face model card. Accessed: 2026-08-05.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36.

A Leakage Audit

All 61 redacted theorem seeds across the 12 paper sets underwent a post-redaction leakage audit. Of these, 58 seeds (95.1%) were flagged for at least one ambiguity or potential leakage concern. A flag indicated a residual inferential risk requiring review or mitigation rather than a failed redaction. After mitigation, all 61 seeds were retained. The audit achieved 100% processing yield, with no skipped files, unmatched outputs, or input–output identifier mismatches, and no audit summary reported an explicit hidden theorem statement or final conclusion remaining in a seed.

The audit recorded 62 concern groups. The most common involved retained proof, algorithm, or construction architecture (40.3%), followed by hints about equivalence, containment, equality, or characterization (21.0%), quantitative formulas or thresholds (14.5%), and structural or definitional setup that narrowed the likely result (12.9%). Mitigation included removing explicit conclusions and quantitative answers, weakening proof dependencies, reframing revealing relationships as open questions or background, and separating definitions from their consequences. Thus, the remaining leakage risk is primarily inferential and contextual rather than explicit.

B Potential Generalization Axes Audit

Because Potential Generalization Axes were synthesized by the same model used for conjecture generation, we conducted a separate content-flow audit to assess whether this component could reintroduce information about the hidden theorem. Each seed's axes were compared with the corresponding exact theorem statement and assigned one seed-level severity:

- **Direct theorem-shaped**: an axis names substantially the same objects, conditions, and conclusion or formula pattern, such that converting the question into a declarative statement approximately reconstructs the theorem.
- **Strong directional**: the axes reveal the theorem's type, architecture, relationship, or expected output, but do not reproduce the complete statement.
- **Low/contextual**: the axes vary assumptions, parameters, methods, or ambient settings without identifying the target conclusion.

Risk level	Seeds	Share
Direct theorem-shaped	12	19.7%
Strong directional	24	39.3%
Low/contextual	25	41.0%
Total	61	100.0%

Table 4: Audit of Potential Generalization Axes for theorem-shaped information.

The 61 seeds contained 450 generalization-axis bullets, with a mean of 7.38 bullets per seed. Table 4 reports the seed-level classifications.

Thus, 36 seeds (59.0%) contained at least strong directional information, including 12 whose axes approximately encoded the shape of the source theorem. This audit characterizes information flow from the source theorem into the constructed component; it does not measure how frequently a generation model would actually reconstruct the theorem from that information.

C Exploratory GLM-5.1-FP8 Judge Analysis

We piloted GLM-5.1-FP8 as an additional expected-theorem-alignment judge on the subset used for the repeated-Gemma and GPT-5-mini analyses. Its scores showed weaker agreement with Gemma, with $r = .649$ and $\mathrm{ICC}(A, 1) = .307$.

An inspection of cases with high Gemma-4-31B and GPT-5-mini scores but low GLM-5.1-FP8 scores suggested that GLM adopted a stricter interpretation of the evaluation prompt. Its rationales frequently noted that the generated output introduced claims "not present in or supported by the Expected Output," effectively treating the expected theorem as the evidentiary basis for every claim. Gemma-4-31B and GPT-5-mini were more permissive toward additional claims when the output otherwise corresponded closely to the target theorem.

We therefore treat the GLM-5.1-FP8 run as an exploratory diagnostic of judge and prompt sensitivity rather than part of the primary cross-model agreement analysis. This post hoc treatment was not based on a prespecified agreement threshold.

D Theorem-Conditioned Conjecture Quality Prompt

- "Evaluate the actual output as a generated mathematical conjecture. The goal is to assign a score based on the quality, relevance, and mathematical usefulness of the conjecture."

- "Use the input as the context that was provided to the conjecture-generating model. Use the expected output as a theorem from a mathematics paper whose preceding text was used to generate the conjecture. Use the actual output as the generated conjecture being evaluated."

- "First, compare the actual output to the expected output. Determine whether the actual output is essentially the same as the expected theorem, a close variant, a related generalization or specialization, loosely related, or unrelated."

- "Second, remove fluff. Ignore generic motivational language, vague claims of importance, broad exposition, future-work language, and text that does not assert a mathematical claim. Determine whether a clear mathematical statement remains."

- "Assess whether anything mathematically meaningful remains after removing fluff. If the remaining content does not contain a clear mathematical statement, assign a low score. If it appears to contain a potentially new mathematical statement, continue evaluating it using the knownness, simple-case plausibility, and relevance criteria below."

- "Assess simple-case plausibility. Penalize statements that are internally inconsistent, obviously false, missing necessary hypotheses, overgeneralized in a way that would likely fail immediately, or likely to fail on the first interesting example. Do not perform extensive computation; use basic mathematical reasoning only."

- "Assess knownness and novelty. A known result is less valuable as a new conjecture than a plausible new statement. However, do not overly penalize close recovery of the expected theorem: similarity to the expected theorem is evidence that the output is mathematically meaningful, interesting, and contextually relevant."

- "Assess relevance to the expected theorem. Outputs close to the expected theorem should generally receive higher scores. Outputs very different from the expected theorem should often receive C-level scores if they contain substantive mathematical content but are difficult to evaluate confidently."

- "Assign a continuous score from 0 to 1 using this scale: 1.0 means A-level: clear, substantive, plausible, highly relevant, and mathematically interesting; it closely recovers, usefully refines, or interestingly extends the expected theorem. 0.75 means B-level: good mathematical conjecture; mostly clear, plausible, and relevant, but less precise, less novel, or less strong than A. 0.5 means C-level: contains potentially interesting mathematical content, but its relevance, novelty, precision, or plausibility is uncertain; this includes many outputs that are quite different from the expected theorem but not clearly bad. 0.25 means D-level: weak conjecture; little substance remains after removing fluff, or the statement is vague, trivial, mostly known, poorly connected, or mathematically imprecise. 0.0 means E-level: no usable conjecture; incoherent, unrelated, essentially only fluff, or obviously false in simple cases."

- "Use intermediate values when appropriate. For example, a score of 0.6 may indicate an output between C and B, while a score of 0.9 may indicate a very strong output that is slightly below ideal."

E Pipeline Prompt Protocol

The conjecture-generation pipeline used three prompts in sequence: (1) theorem classification and theorem-specific object-map construction, (2) theorem-specific redaction, and (3) blind conjecture generation. This appendix summarizes the substantive instructions that governed model behavior. We omit administrative instructions concerning directory paths, filenames, Markdown formatting, identifier bookkeeping, summary files, and behavior when no input files were present.

E.1 Theorem Classification and Object-Map Construction

The first prompt instructed the model to inspect a paper, classify its theorem-like environments, and identify its main proved theorems. The number of qualifying results was not specified in advance, and the model was explicitly instructed not to force a paper to contain a fixed number of main theorems.

A result qualified as a *main proved theorem* only when it satisfied all of the following criteria:

1. it was presented as a contribution of the current paper;
2. it had a proof in the current paper;
3. it was not merely quoted from prior work;
4. it was not a corollary, lemma, proposition, example, remark, definition, or background result;
5. it played a central role in the paper's stated contribution.

Evidence could include emphasis in the abstract or introduction, language such as "we prove," and the presence of a proof in the paper. The model separately classified quoted prior theorems, supporting results, corollaries, and unproved or deferred theorems. Ambiguous cases were to be marked as unclear rather than treated as main proved theorems.

For each qualifying result, the model constructed a separate theorem-specific object map. The object map contained:

- the exact source theorem, its location, and its proof status;
- mathematical objects relevant to the result;
- required definitions and notation;
- assumptions and structural conditions;
- supporting lemmas, propositions, examples, constructions, quoted results, and background facts;
- an explicit or inferred proof-dependency chain;
- a concise proof-strategy summary;
- notes identifying material that should later be removed or generalized.

An object could appear in more than one map when it was relevant to multiple theorems. At this stage, theorem statements and proof information were permitted because the maps served as unredacted intermediate representations. The model was instructed to preserve the source notation, use only the supplied paper, avoid web search, and not invent theorem statements, labels, or proof dependencies.

E.2 Theorem-Specific Redaction

The second prompt converted each object map independently into a redacted seed for blind conjecture generation. Its core instruction was: *keep setup, remove answers*. The seed was intended to represent the local mathematical setting surrounding one source theorem without revealing the theorem itself.

The prompt permitted the seed to retain or nonconclusively rewrite:

- the ambient mathematical setting;
- definitions and notation;
- fields, rings, groups, spaces, graphs, categories, probability spaces, and other domain objects;
- assumptions on parameters;
- relevant subsets, maps, operators, relations, functions, measures, metrics, bases, generators, and constructions;
- hypotheses and broad structural constraints;
- examples and auxiliary objects that did not reveal the conclusion;
- broad, setup-grounded directions along which conjectures might be formed.

The prompt required complete removal of:

- the source theorem statement, conclusion, label, and any revealing title;
- statements describing what the theorem proves;
- the number or identity of the paper's main proved theorems;
- proof summaries, proof sketches, dependency chains, and exact logical paths leading to the theorem;
- result classifications and descriptions identifying a result as main, proved, central, or the paper's primary goal;
- theorem-concluding classifications, decompositions, equivalences, isomorphisms, bounds, rates, formulas, or existence claims;
- descriptions whose purpose was to explain an object's role in the theorem conclusion.

When theorem-dependent text also contained useful setup information, the model was instructed to rewrite it as broader, non-conclusive context rather than copy it. For example, a claim that an invariant equals a particular formula could be reduced to the statement that the setup contains an invariant associated with the objects under study. If information might reveal the conclusion, the prompt instructed the model to remove or generalize it and to prefer under-inclusion over leakage.

Each resulting seed was organized into the six context components used in the ablation study:

1. Ambient Setting;
2. Core Definitions and Notation;
3. Local Mathematical Objects;
4. Structural Constraints and Conditions;
5. Auxiliary Objects and Constructions;
6. Potential Generalization Axes.

The Potential Generalization Axes were to remain generic and grounded in the redacted setup. Suggested directions included varying parameters, weakening or strengthening assumptions, changing the ambient object or target class, replacing a structure with an analogue, and examining small or exceptional cases.

The redaction output also contained a forbidden-content section and a redaction audit checking whether theorem statements, conclusions, proof information, exact classifications, and result-count information had been removed. Before conjecture generation, these two administrative sections were manually removed, leaving only the six mathematical-context components as candidate model input.

E.3 Blind Conjecture Generation

The third prompt instructed the model to generate exactly one plausible formal conjecture for each redacted seed. Under an ablation condition, the model received only the context components retained by that condition. It was not permitted to consult the original paper, LaTeX source, theorem map, unredacted object map, theorem statement, proof summary, dependency chain, redaction report, or any external source.

For each seed, the model was instructed to choose the single most natural conjecture suggested by the available local objects and constraints. The conjecture was written in LaTeX and could use definitions, notation, objects, assumptions, examples, and structural constraints present in the seed. The prompt permitted natural mathematical extrapolation, including a classification, generalization, strengthening, weakening, analogue, exceptional-case statement, structural claim, existence or uniqueness claim, or dimension or bound conjecture. A newly proposed formula was permitted only when it was derived from the redacted setup rather than copied from an unredacted result.

The prompt stated that proximity to the hidden theorem was acceptable, and even desirable, provided that the conjecture followed only from the redacted seed. At the same time, the model was prohibited from claiming that the paper proved the conjecture, claiming knowledge of the hidden theorem, copying an exact conclusion from an unredacted source, or generating multiple candidate conjectures for one seed. Information from one theorem-specific seed could not be used for another unless it appeared independently in both.

For each seed, the requested response contained one precise formal statement, the seed objects and assumptions used, a conjecture-type classification, a plausibility rationale, a tentative proof direction, a counterexample-search strategy, an assessment of possible proximity to a hidden source result, and a blindness and leakage audit. The formal statement was treated as the generated conjecture in the subsequent evaluation.

F Complete Ablation Results

Table 5 reports the complete leave-one-out and single-component results.

G Complete Domain-Level Results

Table 6 reports the complete equal-weight paper-level domain summaries for both evaluation metrics, including all leave-one-out effects. The main-text summary in Table 2 reports only the largest alignment loss and strongest standalone component within each domain. Each domain contains three papers, so these comparisons are exploratory.

(a) Leave-one-out conditions

Metric	Removed component	Full	Ablated	Δ	95% CI	d_z	p_{Holm}
Quality	Ambient setting	.931	.923	-.008	[-.028, .011]	-.108	1.000
	Definitions and notation	.931	.931	.000	[-.022, .022]	.000	1.000
	Local mathematical objects	.931	.923	-.008	[-.030, .013]	-.097	1.000
	Structural constraints	.931	.910	-.021	[-.051, .008]	-.187	1.000
	Auxiliary objects	.931	.938	.007	[-.013, .026]	.085	1.000
	Generalization axes	.931	.936	.005	[-.012, .022]	.074	1.000
Alignment	Ambient setting	.607	.582	-.025	[-.084, .035]	-.106	1.000
	Definitions and notation	.607	.572	-.034	[-.110, .042]	-.116	1.000
	Local mathematical objects	.607	.575	-.031	[-.105, .042]	-.108	1.000
	Structural constraints	.607	.520	-.087	[-.154, -.020]	-.333	.055
	Auxiliary objects	.607	.579	-.028	[-.109, .053]	-.088	1.000
	Generalization axes	.607	.641	.034	[-.038, .107]	.121	1.000

(b) Single-component conditions

Component retained	Quality	Alignment	Alignment gap
Ambient setting	.885	.348	-.259
Definitions and notation	.889	.397	-.210
Local mathematical objects	.898	.457	-.149
Structural constraints	.851	.348	-.259
Auxiliary objects	.844	.270	-.336
Generalization axes	.820	.266	-.341

Table 5: Complete ablation results. Panel (a) reports theorem-level leave-one-out comparisons, and Panel (b) reports standalone performance under the single-component conditions.

(a) Theorem-conditioned conjecture quality

Domain	Full	Ambient	Definitions	Local objects	Constraints	Auxiliary	Generalization
`math.AC`	.934	-.030	-.008	-.018	-.008	.010	-.003
`math.AG`	.881	.000	.029	.019	-.037	.000	.016
`math.CO`	.950	.006	-.019	-.006	-.008	.017	.022
`math.ST`	.986	-.018	-.008	-.059	-.014	-.020	-.014

(b) Expected-theorem alignment

Domain	Full	Ambient	Definitions	Local objects	Constraints	Auxiliary	Generalization
`math.AC`	.611	-.037	-.061	.004	-.162	.011	-.047
`math.AG`	.571	-.052	-.053	-.060	-.121	-.087	-.046
`math.CO`	.644	-.003	-.086	-.042	-.047	.019	.103
`math.ST`	.664	.026	.073	-.096	-.011	-.054	.162

Table 6: Equal-weight paper-level results by domain. The full-context column reports the domain baseline; the remaining columns report leave-one-out differences from that baseline. Negative values indicate lower performance after removing the component. Each domain contains three papers.

The Connective *but* in Informal Mathematical Proofs: A Corpus-Based Study

Rinta Yamada[1,2], Sakiko Yamasaki[2], Risako Ando[2], Takayuki Amamoto[2], Ryota Akiyoshi[1,2], Koji Mineshima[2]

[1]The University of Electro-Communications, Chofu, Tokyo, Japan
[2]Keio University, Minato, Tokyo, Japan
{rinta.yamada.cs,sakiko.yamasaki8008,georg.logic}@gmail.com
{risakochaan,amamoto,minesima}@keio.jp

Abstract

This study presents a focused case study of the explicit connective *but* in informal mathematical proofs. We propose an annotation scheme and analyze 261 occurrences found in 225 proofs from textbooks in logic, set theory, arithmetic, Boolean algebra, and formal semantics. The P-Label distinguishes proof-relevant content—material that forms an essential part of the proof—from metatextual comments on the proof, with contradiction-completing cases marked separately within the proof-relevant class. An optional R-Label records contrast or exception. Of the 261 occurrences, 221 (84.7%) were classified as proof-relevant. Three illustrative case studies with corresponding Lean formalizations show that informal transitions may align with formal operations but need not correspond to single proof lines, clarifying why proof relevance is not defined by line-level correspondence. In grouped five-fold prediction experiments, GPT-5.6 Terra attains 58.6% hierarchical accuracy from the target sentence. In these fixed runs, full-proof input yielded lower hierarchical accuracy in all four comparisons, with observed decreases of 15.3–26.1 points for Qwen3-8B. The study characterizes one connective rather than the full inventory of discourse relations in mathematics, providing a basis for future comparisons with other explicit connectives and implicit relations in informal mathematical proofs.

1 Introduction

Informal proofs in mathematical textbooks and research papers are written in a hybrid language that combines natural language with mathematical notation, exhibiting domain-specific vocabulary and syntactic structures (Ranta, 1995; Ganesalingam, 2013). Beyond the level of sentences, recognizing the relationships between them — namely, the *discourse relations* in mathematical reasoning — is crucial for understanding how arguments are structured and how theorems are justified. However, discourse relations specific to mathematical proofs have received relatively little attention in computational linguistics or in other related fields.

Mathematical discourse employs logical connectives such as *and*, *or*, and *if*, inferential markers such as *therefore*, *since*, and *because*, and other expressions that help readers follow the organization of a proof. Among the latter, *but* is a useful case for examining how an explicit lexical marker interacts with mathematical reasoning. While concession has been studied extensively in theoretical linguistics and discourse analysis (Zufferey and Degand, 2024), corpus-based research on *but* in proof discourse remains limited. Understanding this structure is not only of linguistic interest but also crucial for advancing research on automated proof formalization (Szegedy, 2020; Weng et al., 2025) and informalization (Kaliszyk et al., 2017; Asperti et al., 2026), where the goal is to reconstruct formal proofs from informal mathematical texts and vice versa.

In this study, we propose an annotation scheme and a corpus-based analysis of mathematical textbooks, focusing on the connective *but* in informal mathematical proofs. The study is deliberately framed as a case study of this single word form. We analyze explicit tokens of *but*/*But*; we do not annotate relations expressed by other connectives, compare connective frequencies, or recover implicit discourse relations without an overt marker. Through this focused analysis, we aim to characterize the functions of *but* in mathematical reasoning and establish a basis for comparative studies.

The main findings of this study can be summarized as follows.

First, we construct and annotate a focused corpus of 261 occurrences of *but* in 225 proofs from six mathematical textbooks. We introduce a hierarchical scheme that separates the proof-level contribution of the associated material from optional contrastive and exceptive uses (Sections 3 and 4.1).

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 152–166
October 29, 2026 ©2026 Association for Computational Linguistics

Second, distributional and qualitative analyses show that 221 occurrences are proof-relevant and that 158 of these carry no R-Label, while individual cases illustrate diverse proof-advancing functions of *but* (Sections 4.1 and 4.2).

Third, three illustrative case studies with corresponding formalizations in the proof assistant Lean[1] illustrate two relatively direct alignments and one mismatch, clarifying why line-level alignment does not define proof relevance (Section 4.3).

Finally, grouped five-fold experiments with a linear baseline, GPT-5.6 Terra, and the open-weight Qwen3-8B model provide initial benchmarks for predicting the hierarchical labels and test whether a full proof is more useful than the target sentence alone (Section 5).

2 Discourse Relations and Informal Mathematical Proofs

2.1 Discourse Relations in Natural Language

Discourse relations have been modeled and annotated in frameworks such as Rhetorical Structure Theory (RST) (Carlson et al., 2002), Segmented Discourse Representation Theory (SDRT) (Asher and Lascarides, 2003), and the Penn Discourse Treebank (PDTB) (Prasad et al., 2014; Webber et al., 2019). While RST and SDRT model rhetorical or discourse-semantic structure, PDTB takes an argument-based approach: it identifies explicit or implicit relations between two discourse units, *Arg1* and *Arg2*, and assigns a relation label.

While the PDTB framework characterizes how discourse units are related, informal mathematical proof raises a complementary proof-level question: whether the material associated with a connective supplies a step required to complete the proof. A local relation alone does not record this contribution to the derivation as a whole. Example (1), adapted from Kocurek and Pavese (2022, p. 415), illustrates this distinction.

(1) Suppose p is the largest prime, and let q be a prime divisor of $p! + 1$. Then $q > p$, since no prime at most p divides $p! + 1$. **But** this contradicts the assumption that p is the largest prime. Therefore, there is no largest prime number.

Here the clause introduced by *But* completes the contradiction needed to reject the initial assumption, a proof-level contribution not recorded by a local relation alone. We distinguish an optional relational-use subtype associated with *but* (R-Label) from the proof-level contribution of its associated material (P-Label). Section 3 defines these two annotation dimensions.

2.2 Mathematical Discourse and Formalization

Formal and computational studies of mathematical language have drawn on formal grammar and formal semantics such as Discourse Representation Theory to analyze linguistic and discourse phenomena in mathematical texts (Ranta, 1995; Ganesalingam, 2013; Matsuzaki et al., 2017). The Naproche project similarly uses a controlled natural language and logical representation to support the verification of natural-language proofs (Cramer et al., 2009). Together, these approaches demonstrate the importance of linguistic structure for recovering mathematical content from text.

Their use of discourse representations, however, should be distinguished from annotating the rhetorical organization of an informal proof. These approaches are not primarily designed to record how an overt connective guides a reader through the progression of a proof, or whether its associated material belongs to the derivation rather than to the surrounding exposition. This organizational contribution may disappear in formal representation because a connective such as *but* need not have a direct formal counterpart.

The present study addresses this intermediate layer between discourse-relation annotation and formal proof representation. It distinguishes the proof-level contribution of the material associated with *but* from any local contrastive or exceptive meaning. This perspective complements formalization: the connective may signal an inferential transition even when it does not correspond to a single line or expression in a formal proof.

3 Annotation Scheme for *but*

3.1 Label Definition and Examples

This section introduces a hierarchical annotation scheme for occurrences of the connective *but* in proofs. At the primary level, a Proof-Relevance Label (P-Label) classifies the role of the material associated with *but* in the proof. At the optional second level, a Relation Label (R-Label) records a relational-use subtype when one is identified. Every instance therefore receives a P-Label, but re-

[1] https://lean-lang.org/

ceives an R-Label only when it expresses CONTRAST or EXCEPTION.

3.1.1 Primary Proof-Relevance Labels

At the primary level, we define *Proof-Relevance Labels* (*P-Labels*, for short): PROOF, PROOF-NEG, and META. The P-Label concerns the content associated with *but*, not the word itself: PROOF and PROOF-NEG jointly mark proof-advancing transitions, whereas META marks content that does not advance the derivation. Beyond distinguishing contradiction-completing transitions as PROOF-NEG, we do not annotate finer transition types.

The label PROOF is assigned when the content associated with *but* forms part of the reasoning used to establish the conclusion of the proof.

(2) **Label:** PROOF
[...] we can conclude that $a \in B$. **But** $a \notin B \setminus C$, so it follows that $a \in C$. (Velleman, 2019, p. 111)

The label PROOF-NEG is a more specific proof-relevant category. It is assigned when the associated content states or completes a contradiction that is used to reject an assumption or derive a negated conclusion.

(3) **Label:** PROOF-NEG, CONTRAST
Suppose $x^2 + y = 13$ and $y \neq 4$. Suppose $x = 3$. [...] **But** this contradicts the fact that $y \neq 4$. Therefore $x \neq 3$. (Velleman, 2019, p. 104)

These proofs would become logically incomplete if the content of the clauses initiated by *But* were removed. Note that this deletion test concerns the clause content, not deletion of the connective alone. In particular, the latter clause states the contradiction used to derive the negated conclusion. Thus, (2) is marked with the label PROOF, and (3) has the label PROOF-NEG.

On the other hand, a proof may contain *but* that is used in such contexts as additional remarks, qualifications, caveats, or side comments that are not essential for its validity. We classify this use with the label META: after removing the associated comment, the remaining propositions and inference steps still suffice for the proof. In (4) below, the *but*-clause does not contribute a necessary step to the proof.

(4) **Label:** META
A function $f : \omega \to A$ for which $f(n^+) < f(n)$ (for all $n \in \omega$) is sometimes called a descending chain, **but** this terminology should not be confused with other uses made of the word "chain." (Enderton, 1977, Proof of Theorem 7B, p. 173)

Annotators apply these decisions in the order META, PROOF-NEG, and PROOF; the full decision procedure is given in Appendix A.

3.1.2 Optional Relational-Use Subtypes

The two representative uses of the connective *but* in English are *contrast* and *exception* (Huddleston and Pullum, 2002, p. 1310).

(5) John is tall **but** Paul is short. (Contrast)

(6) Everyone **but** him came. (Exception)

At the optional second level of the hierarchy, we use two labels for relational-use subtypes associated with *but*, namely, CONTRAST and EXCEPTION. These *Relation Labels* (*R-Labels*, for short) supplement the primary P-Label rather than forming an independent classification. The two levels encode different dimensions and are not mutually exclusive alternatives. The P-Label records the proof-level status of the material associated with *but*, whereas the R-Label optionally records whether the same occurrence also instantiates CONTRAST or EXCEPTION. Thus, an occurrence may advance the proof and simultaneously express a contrast, yielding PROOF, CONTRAST.

The term R-Label is used for convenience: the two subtypes are not homogeneous discourse relations. CONTRAST normally relates two discourse arguments, whereas EXCEPTION also includes phrasal exceptives such as (6), which need not connect two clauses and therefore fall outside the usual inventory of PDTB-style explicit discourse relations. We include denial-of-expectation and concessive readings under CONTRAST. Although Contrast and Concession are often distinguished (Kehler, 2002; Winterstein, 2012), including in PDTB 3.0 (Webber et al., 2019), their boundary is difficult to apply consistently in practice; we therefore do not annotate Concession separately. No correction uses, which are most characteristic of dialogue, were found in the present corpus of expository mathematical proofs. Descriptions such as goal shift or inferential continuation concern the proof-functional dimension represented by the P-Label; they are not alternatives to CONTRAST or EXCEPTION and may co-occur with an R-Label.

The CONTRAST label is assigned when the material surrounding *but* explicitly aligns two alternatives and highlights an opposition or difference between them. Both alternatives must be identifiable from the text.

The EXCEPTION label is assigned only when a generalization or shared frame remains in force while a particular member, case, parameter, or respect is excluded or replaced. The occurrence must permit a meaning-preserving paraphrase with *except* or *except for*, as in "the same proof except for the choice of parameter."

(7) **Label:** PROOF, CONTRAST
If E had an element not in D, the complement of such an element would be an upper bound of D **but** not of E. (Halmos, 1974, Proof of Lemma 1, p. 61)

(8) **Label:** META, EXCEPTION
This is the same as the pigeonhole principle, **but** for an arbitrary finite set A instead of a natural number. (Enderton, 1977, Proof of Theorem 6C, p. 135)

Example (7) contrasts the upper-bound status of D and E. In Example (8), *but* marks the parameter that changes within an otherwise shared proof pattern and allows an *except for* paraphrase; we therefore assign CONTRAST and EXCEPTION, respectively.

In informal mathematical proofs, *but* is also commonly used in a way that does not meet either operational R-Label criterion. Consider the following example:

(9) **Label:** PROOF
Suppose $\neg R$. Suppose P. Since P and $P \to (Q \to R)$, it follows that $Q \to R$. **But** then, since $\neg R$, we can conclude $\neg Q$. Thus, $P \to \neg Q$. Therefore $\neg R \to (P \to \neg Q)$. (Velleman, 2019, p. 110)

This occurrence advances the deduction and therefore receives the P-Label PROOF. However, it does not meet either operational R-Label criterion: explicit opposing properties P_1 and P_2 are difficult to identify, and *but* cannot be paraphrased as *except*. Hence no R-Label is added. Proof progression is not an alternative to Contrast: an occurrence that both advances the deduction and expresses an identifiable opposition receives PROOF, CONTRAST, as in Example (7).

3.1.3 Supplementary Annotation

In addition to the labels defined above, each occurrence of *but* was further annotated for its position and form. Occurrences of capitalized *But* at the beginning of a sentence were labeled INITIAL, while lowercase *but* appearing within a sentence was labeled MEDIAL.

Compound expressions with *but* (e.g., *but then*, *but since*) were recorded separately.

3.2 Corpus Overview

The corpus is based on six textbook sources spanning several domains, including logic, set theory, arithmetic, Boolean algebra, and formal semantics. Logic, set theory, and arithmetic are represented in Velleman (2019), Partee et al. (1990), Enderton (2001), and Enderton (1977); Boolean algebra in Halmos (1974) and Partee et al. (1990); and formal semantics in Landman (1991).

To construct the corpus, we identified mathematical statements, including exercise problems, in searchable electronic versions of the sources and searched their associated proof passages for the word forms *but* and sentence-initial *But*. The search covered the full set of target pages in each source. For each match, the corresponding statement and complete proof were OCR-transcribed from the source pages and manually checked against the originals. Proof length was measured over the 225 unique proofs, rather than the 261 target occurrences, using these manually checked OCR transcriptions. We counted Unicode word-like tokens after whitespace normalization and correction of OCR line-break hyphenation; punctuation and standalone mathematical operators were excluded. The median proof length was 124 tokens (IQR 60–196; range 14–696).

Across these sources, 225 proofs containing at least one occurrence of *but* were identified, totaling 261 instances. When multiple occurrences of *but* appeared within a single proof, each was indexed according to its order of appearance and assigned an individual label. Occurrences of *but* in statements were excluded from the count, so only those appearing within proofs were annotated. The extraction targeted only the word forms *but* and sentence-initial *But*. Other connectives (e.g., *and*, *however*, and *although*) and discourse relations without an overt connective were not collected or annotated. Their exclusion reflects the case-study design and should not be interpreted as evidence

P-Label	Contrast	Exception	No R-Label	Total
PROOF	41	5	117	163
PROOF-NEG	17	0	41	58
META	11	5	24	40
Total	**69**	**10**	**182**	**261**

Table 1: Cross-distribution of optional relational-use subtypes and proof-relevance labels.

that they are absent or unimportant in mathematical proofs.

Annotation procedure. Six annotators, all trained in logic, worked in varying pairs, with two independently labeling each instance after inspecting the complete proof. During this phase, they could record an *uncertain* or *other* response for difficult cases. All disagreements and flagged cases were then reviewed by the six annotators, and every instance received an adjudicated P-Label and a decision on whether an R-Label applied. Agreement was calculated from the independent responses, whereas the distributional analyses and prediction experiments use the adjudicated labels for all 261 instances.

Inter-annotator agreement. Because annotator pairs varied across instances, we report Fleiss' κ, based on the pooled category distribution across the six annotators. In the primary analysis, *uncertain* and *other* responses were retained as distinct annotation outcomes rather than removing difficult cases. Agreement was 210/261 (80.5%; $\kappa = .615$) for the P-Label and 176/261 (67.4%; $\kappa = .554$) for the complete hierarchical label, where a case was represented either by its P-Label alone or by the P- and R-Labels together.

R-Label reliability was also evaluated separately. Among the 257 cases with resolved R responses, agreement was 220/257 (85.6%; $\kappa = .661$) on whether to assign any R-Label and 215/257 (83.7%; $\kappa = .636$) on the exact three-way decision among CONTRAST, EXCEPTION, and no R-Label. Restricting the latter analysis to the 210 cases with matching valid P-Labels (209 after excluding one unresolved R response) yielded 176/209 agreement (84.2%; $\kappa = .643$). Appendix A gives sensitivity analyses, the R-Label confusion matrix, an audit of the 10 adjudicated EXCEPTION cases, and qualitative disagreement patterns.

4 Annotation Results and Discussion

4.1 Distribution

We report the adjudicated distribution of all 261 instances from the six textbooks.[2] Table 1 cross-tabulates the primary P-Labels and optional R-Labels.

As shown in Table 1, 221 occurrences (84.7%) were proof-relevant (PROOF or PROOF-NEG) and thus mark proof-advancing transitions in our coarse-grained sense; META uses accounted for 15.3%. Within the proof-relevant category, PROOF cases (163/221; 73.8%) outnumbered PROOF-NEG cases (58/221; 26.2%), showing that general PROOF cases were more common than contradiction-completing PROOF-NEG cases.

The same table shows the distribution of optional relational-use subtypes. The CONTRAST R-Label accounted for 26.4% (69 instances) and EXCEPTION for 3.8% (10 instances). Notably, 69.7% of the instances (182 in total) received no R-Label and were therefore represented by their P-Label alone. Of the 221 proof-advancing transitions, 158 (71.5%; 60.5% of all occurrences) were P-only, carrying neither a CONTRAST nor an EXCEPTION R-Label.

The No R-Label outcome does not mean that these occurrences are unclassified; it means only that neither optional relational-use subtype was identified. A finer-grained pragmatic inventory might subdivide some of these cases, but such distinctions would be orthogonal to the proof-functional classification and would not alter whether the associated content advances the proof. The recurring P-only patterns examined below therefore support the importance of proof-level discourse functions, although comparison with non-mathematical expository discourse would be needed to establish their mathematical specificity.

Table 2 reports the number and median length of the unique proofs in each source, together with the source-level label distribution. Source-level median proof lengths range from 98 tokens in Landman to 161 in Enderton's set-theory text. Proof-relevant uses form the majority in every source, but the balance of R-Labels varies. Except in the seven-instance Halmos sample, receiving no R-Label is the most frequent outcome in the R dimension

[2] Data, annotations, code, and evaluation results are available at https://github.com/kmineshima/but-in-mathematical-proofs.

	Corpus		P-Label			R-Label			
Source	**Proofs**	**Med. length**	**Proof**	**Proof-neg**	**Meta**	**Contrast**	**Exception**	**No R-Label**	***but* occ.**
Velleman	39	118	36	8	2	7	0	39	46
Enderton (set theory)	65	161	37	19	18	21	5	48	74
Enderton (logic)	46	123	25	8	13	9	2	35	46
Landman	43	98	44	13	4	23	2	36	61
Partee et al.	26	99.5	15	10	2	4	0	23	27
Halmos	6	98.5	6	0	1	5	1	1	7
Total	**225**	**124**	**163**	**58**	**40**	**69**	**10**	**182**	**261**

Table 2: Corpus size, median proof length, and final-label distribution by source. Length is measured in word-like tokens. The **Proofs** and **Med. length** columns summarize the 225 unique proofs; label columns and *but* occurrences use 261 target instances.

for every source; the unusually high proportion of CONTRAST in Halmos should therefore be interpreted cautiously. These source-level differences are descriptive and do not by themselves establish effects of mathematical domain or authorial style.

Proof-advancing functions. The PROOF and PROOF-NEG labels mark proof-advancing functions: in both cases, the material associated with *but* contributes to the derivation as a whole. They may invoke an assumption, definition, or earlier result; open a case; or state a consequence completing the current step. We do not annotate finer transition types, whose correspondence to formal operations depends on the chosen formalization. Moreover, *but* cues this proof-advancing function but does not itself make the associated content proof-relevant: the same contribution could be unmarked or differently marked, and deleting only *but* need not make the proof incomplete. The 158 P-only cases show that proof-advancing material frequently occurs without an identifiable opposition or exception.

Supplementary analyses of sentence position and recurring compound expressions are provided in Appendix B.

4.2 Qualitative Analysis and Discussion

To examine internal variation, we focus on recurrent patterns within the PROOF and META labels.

4.2.1 PROOF Label

The PROOF label covers several inferential forms. The examples below include both P-only cases and cases that also receive an R-Label. For example, in (10), *but* marks the point where the assumption $\forall a \in A(f(a) = g(a))$ is applied, separating the general derivation from the step that invokes the given premise.

(10) **Label:** PROOF
Suppose $\forall a \in A\,(f(a) = g(a))$, and let (a, b) be an arbitrary element of f. Then $b = f(a)$. **But** by our assumption $f(a) = g(a)$, so $b = g(a)$ and therefore $(a, b) \in g$. (Velleman, 2019, Proof of Theorem 5.1.4, p. 232)

Introducing assumptions. In (11), *but* contrasts two possibilities and introduces an alternative assumption that guides the subsequent reasoning.

(11) **Label:** PROOF, CONTRAST
If nothing at all is in $\mathcal{C}$, we are done, **but** suppose some function ψ is in $\mathcal{C}$. (Enderton, 2001, Proof of Theorem 36J, p. 260)

This use is equivalent to an implicit *otherwise*, marking a branching point where a new assumption is introduced.

Gap emphasis. Some occurrences of *but* emphasize a gap between a stated conclusion and its supporting reasoning.

In (12), $x \subseteq y$ is short for $x = y$ or $x \in y$. Thus, the statement "$\alpha + \delta = \beta$ or $\alpha + \delta \in \beta$" appears before the *but*-clause, while the equality $\alpha + \delta = \beta$ is asserted after the clause.

(12) **Label:** PROOF, CONTRAST
[...] we know from Theorem 8D that there is a largest δ for which $\alpha + \delta \subseteq \beta$. **But** equality must hold, for [...]. (Enderton, 1977, p. 235)

There is a certain contrast between "$x \subseteq y$" and "$x = y$", and the proof gap is filled by the subsequent clause beginning with *for*. This usage can be regarded as marking a shift to *backward reasoning*, in which the conclusion is stated first, followed by the supporting grounds, rather than proceeding from premises to conclusion.

Additional examples of these qualitative patterns are provided in Appendix C.

4.2.2 META Label

The META label includes expository material that helps readers follow a proof without contributing a necessary inference. One recurrent subtype presents the current proof goal, typically with expressions such as "we must prove" or "we claim."

(13) **Label:** META, CONTRAST
Obviously, *at most* one of the alternatives $x \subset y$, $x = y$, or $y \subset x$ can hold, **but** we must prove that at least one holds.
(Enderton, 1977, Proof of Theorem 5RA, pp. 113–114)

Here, *but* contrasts "at most one" with the remaining goal of proving "at least one." Because stating that goal guides the exposition but is not itself required for the derivation, the occurrence is labeled META, CONTRAST. Additional META examples are provided in Appendix D.

4.3 Illustrative Comparison with Formal Proof Structure

We present three purposively selected case studies from Velleman (2019), using corresponding code provided by Velleman (2023) for the Lean proof assistant (Avigad and Massot, 2025): two relatively direct alignments in Appendix E and one mismatch below. The cases illustrate possible correspondence patterns rather than their prevalence. Because the mapping depends on the chosen formalization, they clarify why the P-Label criterion cannot require line-level correspondence.

(14) **Theorem.** Suppose B is a set and F is a family of sets. If $\bigcup F \subseteq B$ then $F \subseteq \mathcal{P}(B)$. (Velleman, 2019, p. 125)

Proof. (Velleman, 2019, p. 125)

(L1) Suppose $\bigcup F \subseteq B$.
(L2) Let x be an arbitrary element of F.
(L3) Let y be an arbitrary element of x.
(L4) Since $y \in x$ and $x \in F$, by the definition of $\bigcup F$, $y \in \bigcup F$.
(L5) **But** then since $\bigcup F \subseteq B$, $y \in B$.
(L6) Since y was an arbitrary element of x, we can conclude that $x \subseteq B$, so $x \in \mathcal{P}(B)$.
(L7) **But** x was an arbitrary element of F, so this shows that $F \subseteq \mathcal{P}(B)$, as required.

The following is the corresponding Lean proof given by Velleman (2023, pp. 40–43).

```
theorem Example_3_3_5 (U : Type) (B : Set U)
(F : Set (Set U)) : ⋃₀ F ⊆ B → F ⊆ P B := by
assume h1 : ⋃₀ F ⊆ B
define
fix x : Set U
assume h2 : x ∈ F
define
fix y : U
assume h3 : y ∈ x
define at h1
apply h1 _
define
apply Exists.intro x _
show x ∈ F ∧ y ∈ x from And.intro h2 h3
done
```

The informal proof explicitly marks universal-generalization steps in (L6) and (L7), mainly for pedagogical reasons, while no single Lean line directly corresponds to the *but*-clause in (L5). The formal proof instead completes the same inferential work from its goal and local context. Thus, a *but*-clause can be proof-relevant even without a one-to-one formal counterpart: the P-Label is determined by its role in the informal derivation, not by surface alignment with Lean code.

5 Automatic Label Prediction

5.1 Experimental Setup

We evaluated label prediction using grouped stratified five-fold cross-validation, pooling held-out predictions across folds. Proofs with the same normalized statement were assigned to the same fold, yielding 212 split groups from 225 proof records (261 instances) and preventing closely related proofs from crossing the train–test boundary.

Baselines were the training-fold majority and a class-balanced linear softmax classifier using sentence-level word unigram and bigram TF–IDF features. The linear classifier receives the target sentence with the target occurrence replaced by a special marker, and a separate classifier is trained for each task. Through OpenRouter, we also evaluated GPT-5.6 Terra and the open-weight Qwen3-8B (Qwen Team, 2025), comparing marked target-sentence and full-proof input under zero- and eight-shot prompting. The eight demonstrations represented the eight observed combined labels and were selected deterministically from training-fold instances; none came from a held-out split group. Appendix F gives the full settings; data, predictions, and code are available in the public repository linked in Section 4.1.

For the P-Label, R-Label, and complete hierarchical label, we report accuracy and macro-F1,

			P-Label		R-Label		Hierarchical	
Model	**Input**	**Prompt**	**Acc.**	**F1**	**Acc.**	**F1**	**Acc.**	**F1**
Majority	Sentence	—	.625	.256	.697	.274	.448	.077
TF–IDF linear	Sentence	—	.667	.570	**.797**	**.716**	.579	**.476**
Terra	Sentence	Zero	.751	.708	.747	.569	**.586**	.413
Terra	Sentence	8-shot	.751	.695	.720	.700	.556	.379
Terra	Full proof	Zero	**.759**	**.724**	.628	.615	.498	.363
Terra	Full proof	8-shot	.739	.705	.655	.677	.494	.348
Qwen3-8B	Sentence	Zero	.460	.333	.655	.422	.341	.137
Qwen3-8B	Sentence	8-shot	.682	.574	.602	.559	.456	.314
Qwen3-8B	Full proof	Zero	.241	.143	.318	.200	.080	.029
Qwen3-8B	Full proof	8-shot	.670	.499	.421	.428	.303	.217

Table 3: Pooled held-out results over five grouped folds. F1 is macro-F1. Bold indicates the best value in each metric column.

which gives equal weight to rare labels. For sentence/full-proof accuracy differences, unadjusted percentile 95% CIs were obtained from 20,000 paired bootstrap resamples of the 212 groups; we make no dichotomous significance claims.

5.2 Results and Error Analysis

Table 3 shows that sentence-only zero-shot Terra achieves the highest hierarchical accuracy (.586), only 0.7 points above the TF–IDF baseline (.579), whose macro-F1 is higher (.476 versus .413). Terra therefore does not consistently outperform the linear baseline, especially on rare labels. Eight-shot prompting raises Terra's sentence-level R-Label macro-F1 (.569 to .700) but lowers hierarchical accuracy; for Qwen, it broadens the label set and raises hierarchical accuracy by 11.5 points with sentence input and 22.2 points with full proofs.

In the fixed runs, full-proof input yielded lower hierarchical accuracy in all four comparisons. The full-proof-minus-sentence differences were -8.8 points (95% CI $[-16.0, -1.5]$) and -6.1 points ($[-12.5, 0.4]$) for Terra, and -26.1 points ($[-32.4, -19.7]$) and -15.3 points ($[-21.7, -8.9]$) for Qwen in the zero- and eight-shot conditions, respectively.

Prediction distributions suggest that additional context introduced competing cues despite explicit target markers. Terra zero-shot assigns CONTRAST 84 times from sentences but 149 times from full proofs (69 gold), while Qwen's zero-shot full-proof run assigns PROOF-NEG, CONTRAST to 239 of 261 instances. Among the paired Terra zero-shot predictions, 59 instances are correct only with sentence input, compared with 36 only with full-proof input. For example, for "But $a \notin B \setminus C$, so it follows that $a \in C$," the gold and sentence-only Terra labels are PROOF, whereas the full-proof prediction adds CONTRAST.

Errors also track annotation difficulty: the best Terra condition reaches .670 hierarchical accuracy on the 176 instances with identical pre-adjudication labels, but .412 on the 85 disagreement instances. Frequent errors add CONTRAST to PROOF or confuse PROOF with PROOF-NEG. Exceptive combinations remain difficult: sentence-only zero-shot Terra predicts Exception only four times (10 gold) and never correctly predicts META, EXCEPTION; this helps explain the low hierarchical macro-F1 despite stronger P-Label accuracy.

6 Summary and Prospects

This study constructed and annotated a focused corpus of 261 occurrences of *but* in 225 informal proofs from six textbooks. Its hierarchical scheme separates proof-level contribution from optional contrastive and exceptive uses, treating connective use as part of mathematical reasoning. Of the 261 occurrences, 221 (84.7%) marked proof-relevant transitions. Qualitative analysis and three Lean comparisons further showed that proof relevance cannot be reduced to line-level correspondence with formal proofs. Prediction experiments established initial benchmarks; in the fixed runs, full-proof input yielded lower accuracy in all four comparisons, with Qwen3-8B decreases of 15.3–26.1 points. Together, the corpus, scheme, and analyses provide an empirical basis for studying connectives in informal mathematical language. Future work should extend the analysis to other connectives and implicit relations and examine informal–formal correspondence on larger datasets.

Limitations

The corpus is small and domain-specific: it contains 261 occurrences of *but* from six textbooks and excludes other connectives and implicit discourse relations. The results should therefore be read as a case study of explicit *but*, not as a general inventory or frequency comparison of discourse relations in mathematical proofs. Comparisons with non-mathematical expository discourse are needed before treating the observed P-only patterns as specific to mathematical proof.

The annotation scheme necessarily involves interpretive judgments, particularly for R-Labels and the functionally heterogeneous P-only cases. Resolved disagreements concentrated at the boundary between PROOF and META, the distinction between intermediate proof steps and completed contradictions, and the decision whether an explicit R-Label was warranted. The EXCEPTION subtype remains provisional: only 10 instances received this adjudicated label, and both independent annotators selected it in only five of them.

Both supplementary evaluations are exploratory. The Lean analysis consists of three purposively selected cases—two relatively direct alignments and one mismatch—and illustrates possible correspondence patterns rather than their prevalence. The mapping also depends on the chosen formalization. The prediction benchmark is limited to two LLMs, one linear baseline, two context windows, and one fixed zero-shot or eight-shot prompt per condition, without repeated stochastic runs, prompt optimization, or fine-tuning. Label imbalance makes macro-F1 uncertain. The bootstrap intervals reflect variation across the sampled proof groups, not repeated model runs, alternative prompts, or providers; the observed full-proof differences may therefore depend on prompting.

Acknowledgements

We thank the anonymous reviewers for their helpful comments and suggestions, which have improved this paper. This work was partially supported by JST CREST Grant Number JPMJCR2114, JST BOOST Grant Number JPMJBS2409 and JSPS KAKENHI Grant Number JP24K03340, JP24K00004.

During manuscript preparation, we used OpenAI's Codex to assist with language editing and checks of internal consistency and bibliographic information. The authors reviewed the resulting suggestions and take full responsibility for the final content.

References

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

Andrea Asperti, Alberto Naibo, and Claudio Sacerdoti Coen. 2026. Thinking machines: Mathematical reasoning in the age of LLMs. *Big Data and Cognitive Computing*, 10(1):38.

Jeremy Avigad and Patrick Massot. 2025. Mathematics in Lean. Release v4.19.0. https://avigad.github.io/mathematics_in_lean/mathematics_in_lean.pdf.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2002. *RST discourse treebank*. Linguistic Data Consortium, University of Pennsylvania.

Marcos Cramer, Bernhard Fisseni, Peter Koepke, Daniel Kühlwein, Bernhard Schröder, and Jip Veldman. 2009. The Naproche project controlled natural language proof checking of mathematical texts. In *International Workshop on Controlled Natural Language*, pages 170–186. Springer.

Herbert B Enderton. 1977. *Elements of Set Theory*. Academic Press.

Herbert B Enderton. 2001. *A Mathematical Introduction to Logic*, second edition. Academic Press.

Mohan Ganesalingam. 2013. *The Language of Mathematics: A Linguistic and Philosophical Investigation*. Springer.

Paul R. Halmos. 1974. *Lectures on Boolean Algebras*. Springer.

Rodney D. Huddleston and Geoffrey K. Pullum. 2002. *The Cambridge Grammar of the English Language*. Cambridge University Press.

Cezary Kaliszyk, Josef Urban, and Jiří Vyskočil. 2017. Automating formalization by statistical and semantic parsing of mathematics. In *International Conference on Interactive Theorem Proving*, pages 12–27. Springer.

Andrew Kehler. 2002. *Coherence, Reference, and the Theory of Grammar*. CSLI publications.

Alexander W. Kocurek and Carlotta Pavese. 2022. The dynamics of argumentative discourse. *Journal of Philosophical Logic*, 51:413–456.

Fred Landman. 1991. *Structures for Semantics*. Springer.

Takuya Matsuzaki, Takumi Ito, Hidenao Iwane, Hirokazu Anai, and Noriko H. Arai. 2017. Semantic parsing of pre-university math problems. In *Proceedings of the 55th Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 2131–2141, Vancouver, Canada. Association for Computational Linguistics.

Barbara H. Partee, Alice ter Meulen, and Robert E. Wall. 1990. *Mathematical Methods in Linguistics*. Springer.

Rashmi Prasad, Bonnie Webber, and Aravind Joshi. 2014. Reflections on the Penn Discourse TreeBank, Comparable Corpora, and Complementary Annotation. *Computational Linguistics*, 40(4):921–950.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Aarne Ranta. 1995. Syntactic categories in the language of mathematics. In *Types for Proofs and Programs*, pages 162–182. Springer.

Christian Szegedy. 2020. A promising path towards autoformalization and general artificial intelligence. In *International Conference on Intelligent Computer Mathematics*, pages 3–20. Springer.

Fenner Stanley Tanswell and Matthew Inglis. 2024. The language of proofs: A philosophical corpus linguistics study of instructions and imperatives in mathematical texts. In Bharath Sriraman, editor, *Handbook of the History and Philosophy of Mathematical Practice*, pages 2925–2952. Springer.

Daniel Velleman. 2023. How to prove it with Lean. URL: https://djvelleman.github.io/HTPIwL.

Daniel J. Velleman. 2019. *How to Prove It: A Structured Approach*, third edition. Cambridge University Press.

Bonnie Webber, Rashmi Prasad, Alan Lee, and Aravind Joshi. 2019. The Penn discourse treebank 3.0 annotation manual. Technical report, University of Pennsylvania.

Ke Weng, Lun Du, Sirui Li, Wangyue Lu, Haozhe Sun, Hengyu Liu, and Tiancheng Zhang. 2025. Autoformalization in the era of large language models: A survey. *arXiv preprint arXiv:2505.23486*.

Grégoire Winterstein. 2012. What *but*-sentences argue for: An argumentative analysis of *but*. *Lingua*, 122(15):1864–1885.

Sandrine Zufferey and Liesbeth Degand. 2024. *Connectives and Discourse Relations*. Cambridge University Press.

A Annotation Guidelines

This appendix consolidates the decision rules used for the hierarchical annotation. The target of annotation is one overt token of *but* or *But*; annotators inspect the complete proof containing that token, not only the sentence in which it occurs.

Step 1: assign the P-Label. The P-Label concerns the content associated with *but*, not whether the connective itself can be deleted. The two proof-relevant outcomes, PROOF and PROOF-NEG, jointly operationalize proof-advancing transitions at a coarse level; META does not. Except for distinguishing contradiction-completing transitions as PROOF-NEG, the annotation does not assign a more specific transition type. Annotators apply the following decision order.

1. If removing the associated content leaves the stated propositions and inference steps sufficient for the proof, assign META. Typical cases are terminological cautions, explanatory asides, and comments that improve readability without advancing the derivation.

2. Otherwise, if the content states or completes a contradiction used to reject an assumption or derive a negated conclusion, assign PROOF-NEG.

3. Otherwise, assign PROOF. This includes required assumptions, applications of definitions or earlier results, case distinctions, and consequences that complete a proof obligation.

Because PROOF-NEG is the more specific proof-relevant label, it takes precedence over PROOF whenever its contradiction criterion is satisfied.

Two boundary rules are important. First, mathematical content alone does not warrant PROOF: necessity is evaluated relative to the stated proof obligation, so goal or method announcements and continuations after that obligation has already been discharged may be META. Second, occurring within a reductio argument does not by itself warrant PROOF-NEG. The target content must state or complete the contradiction; an intermediate consequence that merely advances the reductio remains PROOF.

Step 2: assign an R-Label only if applicable. R-Labels are evaluated after the P-Label and independently of proof relevance; for example, both PROOF, CONTRAST and META, CONTRAST are possible. An R-Label is optional and is not an alternative to the P-Label: inferential continuation, goal presentation, or other proof-level functions do not preclude an R-Label when an explicit contrast or exception is also present.

1. Test EXCEPTION first. Assign it only if a generalization or shared frame remains applicable except for an explicitly excluded or substituted member, case, parameter, or respect, and the passage permits a meaning-preserving paraphrase with *except* or *except for*.

2. If the exceptive test fails, test CONTRAST. Assign it when the text explicitly aligns two entities, properties, states, or alternatives and highlights their opposition or difference. The aligned pair must be identifiable from the text; a clause's inferential importance alone is insufficient. Denial-of-expectation and concessive readings are included under CONTRAST; no separate Concession label is assigned.

3. If neither test succeeds, assign no R-Label. Absence of an R-Label is not a third relation category.

The EXCEPTION label includes phrasal exceptive uses and therefore should not be interpreted as a PDTB-style discourse-relation label. Negation, contradiction, or the general adversative force of *but* is not by itself sufficient for CONTRAST; the aligned entities, properties, states, or alternatives must be textually identifiable.

Step 3: record the combined label. The P-Label is written first. A P-only output is written as PROOF; when an R-Label is assigned, the two labels are separated by a comma, as in PROOF, CONTRAST or META, EXCEPTION.

Supplementary fields. Annotators also record whether the token is sentence-initial or sentence-medial and note recurring compound expressions such as *but then*, *but since*, and *but when*. These fields do not determine the P-Label or R-Label.

Uncertainty and adjudication. Annotators completed the two label decisions independently and could record *uncertain* or *other* for difficult cases. These were response options for flagging problems, not labels in the final scheme. After both annotations were complete, all disagreements and flagged cases were discussed by the six annotators; every instance received an adjudicated P-Label and a final decision among CONTRAST, EXCEPTION, and no R-Label. All 261 adjudicated instances were retained in the distributional analyses and prediction experiments.

Agreement calculation. The primary agreement analysis retains *uncertain* and *other* as distinct responses, thereby including rather than selectively removing difficult cases. As a sensitivity analysis, we also excluded any case in which either annotator used one of these responses. P-Label agreement then increased from 210/261 (80.5%; Fleiss' $\kappa = .615$) to 210/254 (82.7%; $\kappa = .647$), while complete hierarchical-label agreement increased from 176/261 (67.4%; $\kappa = .554$) to 176/253 (69.6%; $\kappa = .578$). The numerators are unchanged because all excluded cases were disagreements, and the overall pattern is stable under the alternative treatment.

R-Label reliability. We evaluated R decisions separately after excluding four cases in which either annotator gave an unresolved R response. Agreement was 220/257 (85.6%; $\kappa = .661$) for the binary decision between no R-Label and an assigned R-Label, and 215/257 (83.7%; $\kappa = .636$) for the exact three-way decision among CONTRAST, EXCEPTION, and no R-Label. The two annotators gave matching valid P-Labels in 210 cases; after excluding the one case with an unresolved R response, exact R agreement was 176/209 (84.2%; $\kappa = .643$).

R1 \ R2	Contrast	Exception	No R	Total
CONTRAST	50	4	14	68
EXCEPTION	1	5	9	15
No R-Label	14	0	160	174
Total	65	9	183	257

Table 4: Independent R-Label decisions for the 257 cases with resolved R responses.

Because annotator pairs varied across items, the rows and columns in Table 4 denote annotation slots rather than fixed individuals. When both annotators assigned a subtype, they agreed on CONTRAST versus EXCEPTION in 55/60 cases (91.7%; $\kappa = .619$); the five remaining cases were direct subtype confusions.

Adjudicated EXCEPTION cases. The 10 adjudicated EXCEPTION cases show greater uncertainty than this aggregate rate suggests. Their independent R pairs were EXCEPTION / EXCEPTION (5 cases) and EXCEPTION / no R-Label (5). Thus both annotators independently selected EXCEPTION in 5/10 cases, exactly one did so in 5/10, and neither did so in 0/10. Because this subset is defined by the adjudicated outcome, these counts are treated

as a descriptive audit rather than as the basis for a separate κ.

Disagreement patterns. Among the 44 P-Label disagreements with resolved responses, PROOF / META and PROOF / PROOF-NEG each occurred 21 times, while META / PROOF-NEG occurred twice. Adjudication selected META in 15 of the 21 PROOF / META cases and PROOF-NEG in 17 of the 21 PROOF / PROOF-NEG cases. Eight of the latter 21 cases contained *but then*, which often advances a contradiction argument without determining whether the target sentence itself completes the contradiction.

The following paired examples illustrate why full proof context and mathematical knowledge are needed at the two most frequent P-Label boundaries.

(15) **Independent P-Labels:** META / PROOF
Adjudicated label: META
Since α is a set, [...] some ordinals do not belong to α. This proves the theorem, **but** we will continue and show that α itself is the least such ordinal. (Enderton, 1977, Proof of Hartogs' Theorem, p. 196)

(16) **Independent P-Labels:** PROOF / META
Adjudicated label: PROOF
Assume that $g \circ f$ is not onto. [...] **But** g is onto C, and thus there is some $y \in B$ such that $g(y) = z$. (Partee et al., 1990, Exercise 6, p. 581)

In (15), the stated theorem has already been established, so the following strengthening is META. In (16), surjectivity supplies a witness required for the subsequent contradiction, so the target is PROOF.

(17) **Independent P-Labels:** PROOF / PROOF-NEG
Adjudicated label: PROOF
Thus if A is recursive, then f (as a relation) is recursively enumerable. **But** then f is a total recursive function, and so equals $[\![e]\!]_1$ for some $e \in A$. (Enderton, 2001, Proof of Theorem 36E, p. 254)

(18) **Independent P-Labels:** PROOF / PROOF-NEG
Adjudicated label: PROOF-NEG
[...] the only other possibility is $x = b$. **But** then since xRc we must have bRc, which contradicts our assumption that $\neg bRc$. (Velleman, 2019, Solution to Example 6.2.1, p. 346)

Example (17) is an intermediate consequence; the contradiction appears only in the next sentence. Example (18) explicitly completes the contradiction and is therefore PROOF-NEG.

Table 4 shows that the R-applicability boundary produced 37 disagreements, compared with five direct CONTRAST / EXCEPTION confusions. Of the 37 disagreements over whether to assign any R-Label, adjudication assigned one in 18 cases and none in 19, confirming a genuine boundary. Example (2) illustrates no R-Label: its negative premise advances the proof without aligning opposing alternatives. By contrast, Example (19) aligns properties of the same object, while Example (20) preserves a proof frame but replaces one operation with another.

(19) **Independent R decisions:**
CONTRAST / no R-Label
Adjudicated label: PROOF, CONTRAST
Then there exists some $t \in A$ with $\operatorname{seg} t \subseteq B$ **but** $t \notin B$. (Enderton, 1977, Proof of Theorem 7C, p. 175)

(20) **Independent R decisions:**
EXCEPTION / no R-Label
Adjudicated label: PROOF, EXCEPTION
The proof is exactly as in Lemma 5ZB, **but** with addition replaced by multiplication. (Enderton, 1977, Proof of Lemma 5QD, p. 106)

B Sentence Position and Compound Expressions

Sentence position appears to interact systematically with discourse function. As summarized in Table 5, sentence-initial *But* appeared in 173 out of 261 cases (66.3%), predominantly within proof-relevant contexts (PROOF=103, PROOF-NEG=50). In contrast, sentence-medial *but* occurred mostly in PROOF contexts, though a smaller proportion appeared in META clauses.

Table 6 shows that sentence-initial *But* frequently co-occurred with the absence of an R-Label (150/173 = 86.7%), whereas sentence-medial *but* carried CONTRAST or EXCEPTION in 56 of 88 cases (63.6%). This pattern is consistent with initial *But* serving as a discourse-level transition cue and medial *but* expressing local contrast or exception within proof steps.

Among compound expressions, three combinations were notable: (i) *but then* appeared most frequently (52 instances), all in proof-relevant contexts (PROOF: 33; PROOF-NEG: 19), and received no R-Label in 47 cases; (ii) *but since* occurred 16 times (PROOF: 13; PROOF-NEG: 3) and received no R-Label in 15 cases, usually introducing a step justified by a stated reason; and (iii) *but when* was rare (2 instances), both labeled META with no R-Label.

Label	Initial	Medial	Total
PROOF	103	60	163
PROOF-NEG	50	8	58
META	20	20	40
Total	173	88	261

Table 5: Proof-relevance label by sentence position. Initial denotes sentence-initial; medial denotes sentence-medial.

R-Label	Initial	Medial	Total
CONTRAST	23	46	69
EXCEPTION	0	10	10
No R-Label	150	32	182
Total	173	88	261

Table 6: Optional relational-use subtype by sentence position. Initial denotes sentence-initial; medial denotes sentence-medial.

C Additional Qualitative Examples for the PROOF Label

This appendix supplements the three representative examples retained in Section 4.2.

Applying an inductive hypothesis. In (21), *but* introduces the application of the inductive hypothesis after a step based on the definition of B_{n+1}. It thus cues a shift between adjacent inferential operations rather than an explicit contrast.

(21) **Label:** PROOF
Now suppose $x \in B_{n+1}$. By the definition of B_{n+1} this means $x \in f(B_n)$, so there is some $b \in B_n$ such that $x = f(b)$. **But** by the inductive hypothesis, $B_n \subseteq D$, so $b \in D$, and since D is closed under f it follows that $x = f(b) \in D$. (Velleman, 2019, Proof of Theorem 6.5.1, p. 318)

In (22), *but* introduces the decisive application of the one-to-one property of f, moving from $f(a_1) = f(a_2)$ to $a_1 = a_2$.

(22) **Label:** PROOF
Thus, $a_2 \notin Y$, so $a_2 \in X$. This means that $h(a_2) = f(a_2)$, so from the equation $h(a_1) = h(a_2)$ we get $f(a_1) = f(a_2)$. **But** f is one-to-one, so it follows that $a_1 = a_2$. (Velleman, 2019, Proof of Theorem 8.3.2, p. 392)

Introducing assumptions. As in the example retained in the main text, *but* in (23) marks a branching point and introduces the alternative assumption that S is nonempty.

(23) **Label:** PROOF, CONTRAST
We hope that S is empty. **But** if not, then there is a least $t \in S$.
(Enderton, 1977, Proof of Theorem 7D, p. 183)

Trivial steps. Expressions such as "But this is clear" may introduce an immediate but logically indispensable step and are therefore assigned PROOF.

(24) **Label:** PROOF
We must show that $\mathrm{card}(V_\omega) = \aleph_0$. **But** this is clear, since V_ω is the union of $\aleph_0$ finite sets of increasing size.
(Enderton, 1977, Proof of Theorem 9J, p. 252)

Gap emphasis. In (25), the *but*-clause derives $\mathrm{card}\, S \geq \aleph_{\alpha+1}$ from $\aleph_{\alpha+1} \leq (\mathrm{card}\, S) \cdot \aleph_\alpha$ while leaving the relevant cardinal-arithmetic properties implicit. The connective draws attention to an inferential step for which further explanation could be supplied.

(25) **Label:** PROOF
Therefore [...] we have $\aleph_{\alpha+1} = \mathrm{card} \bigcup S \leq (\mathrm{card}\, S) \cdot \aleph_\alpha$. **But** this implies that $\mathrm{card}\, S \geq \aleph_{\alpha+1}$. (Enderton, 1977, Proof of Theorem 9M, p. 256)

D Subclasses of META Label

This appendix describes subclasses of the META label. These cases typically involve *but* appearing in remarks, qualifications, or side comments within proofs. Although omitting these segments would not affect the validity of the proof, they contribute to clarity and readability and are therefore annotated as META.

Adding remarks. In this subclass, *but* introduces a comment or clarification outside the core inferential steps of the proof.

(26) **Label:** META, CONTRAST
Clearly f is onto, since by definition all rational numbers can be written as a fraction, **but** note that f is not one-to-one.
(Velleman, 2019, Proof of Theorem 8.1.6, p. 458)

(27) **Label:** META
The methods of Sections 3.3 and 3.4 can be applied to the Gödel numbers of these formulas to show that p is recursive. **But** the details are not particularly attractive, and we omit them here.
(Enderton, 2001, Proof of Lemma 37E, p. 273)

Interrogative goal presentation. The following example illustrates the use of an interrogative within a proof, used in presenting the goal of the argument. The presence of such non-declarative sentences demonstrates the diversity of linguistic forms that appear in informal proofs.[3]

(28) **Label:** META
We must show that $\phi(\alpha, y) \Leftrightarrow \gamma(f, y)$ for each y. **But** what is f? It must be the function F_α, by the above.
(Enderton, 1977, pp. 210–212)

E Detailed Lean Comparisons

This appendix presents the two relatively direct alignment cases that complement the mismatch case in Section 4.3. We use the Lean library and tactics provided by Velleman (2023).

A PROOF-NEG example. Consider the following theorem and informal proof.

(29) **Theorem.** Suppose A, B, and C are sets, $A \setminus B \subseteq C$, and x is anything at all. If $x \in A \setminus C$ then $x \in B$. (Velleman, 2019, p. 107)

Proof. (Velleman, 2019, p. 107)

(L1) Suppose $x \in A \setminus C$.
(L2) This means that $x \in A$ and $x \notin C$.
(L3) Suppose $x \notin B$.
(L4) Then $x \in A \setminus B$, so since $A \setminus B \subseteq C$, $x \in C$.
(L5) **But** this contradicts the fact that $x \notin C$.
(L6) Therefore $x \in B$.
(L7) Thus, if $x \in A \setminus C$ then $x \in B$.

The corresponding Lean proof is:

```
theorem Example_3_2_3
  (U : Type) (A B C : Set U) (x : U)
  (h : A \ B ⊆ C )
  (h1 : x ∈ A \ C) :
  x ∈ B := by
define at h1
by_contra h2
have h3 : x ∈ A := h1.left
have h4 : x ∈ A ∧ x ∉ B := And.intro h3 h2
define at h
have h5 : x ∈ C := h h4
show False from h1.right h5
done
```

The theorem declaration contains the assumptions and conclusion, with (L1) represented by the hypothesis `h1`; the next command expands this hypothesis as in (L2). The command `by_contra h2` introduces the negated conclusion corresponding to (L3), after which the proof derives $x \in A \setminus B$ and then $x \in C$, matching (L4). Finally, `show False from h1.right h5` corresponds to the contradiction stated in the *but*-clause in (L5); the remaining informal sentences close the argument.

A PROOF example. The next case also permits a relatively direct comparison.

(30) **Theorem.** There is a unique set A such that for every set B, $A \cup B = B$. (Velleman, 2019, p. 157)

Proof. (Velleman, 2019, pp. 157–158)

(L1) Existence: Clearly $\forall B(\varnothing \cup B = B)$, so $\varnothing$ has the required property.
(L2) Uniqueness: Suppose $\forall B(C \cup B = B)$ and $\forall B(D \cup B = B)$.
(L3) Applying the first assumption to D gives $C \cup D = D$, and applying the second to C gives $D \cup C = C$.
(L4) **But** clearly $C \cup D = D \cup C$, so $C = D$.

The corresponding Lean proof is (Velleman, 2023, pp. 83–87):

```
theorem Example_3_6_2 (U : Type) :
∃! (A : Set U), ∀ (B : Set U), A ∪ B = B := by
exists_unique
· -- Existence
 apply Exists.intro ∅
 show ∀ (B : Set U), ∅ ∪ B = B from empty_union
 done
· -- Uniqueness
 fix C : Set U; fix D : Set U
 assume h1 : ∀ (B : Set U), C ∪ B = B
 assume h2 : ∀ (B : Set U), D ∪ B = B
 have h3 : C ∪ D = D := h1 D
 have h4 : D ∪ C = C := h2 C
```

[3] For a corpus-based analysis of imperative expressions in proofs, such as *Suppose that* and *Let*, see Tanswell and Inglis (2024).

```
  show C = D from
   calc C
   _ = D ∪ C := h4.symm
   _ = C ∪ D := union_comm D C
   _ = D := h3
   done
  done
```

The `exists_unique` tactic divides the proof into existence and uniqueness: the existence block corresponds to (L1), while the assumptions and their applications in the uniqueness block correspond to (L2) and (L3). The final `calc` block uses commutativity of union to conclude $C = D$, matching the inferential role of the *but*-clause in (L4).

F Model Evaluation Reproducibility

Models and inference settings. Both LLMs were queried through the OpenRouter Chat Completions API with JSON Schema output restricted to the eight observed combined labels in Table 1. For Terra, the model identifier was `openai/gpt-5.6-terra`; requests were pinned to Azure with provider fallbacks disabled, reasoning effort set to `low`, temperature left at the provider default, and `max_completion_tokens=256`. For Qwen3-8B (Qwen Team, 2025), the identifier was `qwen/qwen3-8b`; all response metadata identified the Alibaba FP8 endpoint, thinking was disabled, temperature was 0, and `max_tokens=64`. Each condition was run once.

Input construction. The sentence condition supplied only the sentence containing the target token, whereas the full-proof condition supplied the complete proof but not its statement. In both conditions, the relevant token was enclosed in `<TARGET>` and `</TARGET>` tags. No input truncation was applied: full proofs and all eight demonstrations were passed in full. The longest serialized prompt contained 14,137 characters.

The artifact does not redistribute the source statements or complete proof texts because they originate in copyrighted textbooks; it provides the annotations, split metadata, completed predictions, and analysis code needed to audit the reported results.

Complete system prompt. The following reproduces the static system prompt; only typography has been normalized for print.

> You annotate the discourse function of one marked occurrence of "but" in an informal mathematical proof.
>
> Apply the hierarchy in this order.
>
> **Step 1—P-label:**
>
> - Meta: removing the associated content leaves the stated propositions and inference steps sufficient. Typical cases are terminological cautions, explanatory asides, and readability comments.
> - Proof-neg: the content states or completes a contradiction used to reject an assumption or derive a negated conclusion.
> - Proof: the content is otherwise required for the derivation, including assumptions, definitions, earlier results, case distinctions, and consequences completing a proof obligation.
>
> Proof-neg takes precedence over Proof when its criterion holds.
>
> **Step 2—optional R-label, independently of the P-label:**
>
> - Exception: test first. A generalization or shared frame remains applicable except for an explicitly excluded or substituted member, case, parameter, or respect; an "except" or "except for" paraphrase preserves the meaning.
> - Contrast: if Exception fails, assign Contrast only when the text explicitly aligns identifiable entities, properties, states, or alternatives and highlights an opposition or difference.
> - Otherwise assign no R-label.
>
> The target token is enclosed in `<TARGET>` and `</TARGET>`. Classify the associated content, not whether the word "but" itself can be deleted.

User-prompt templates. The zero-shot user message was "Classify this target: [marked input]. Apply Step 1 and Step 2, then return the combined label." Line breaks separated the introductory text, marked input, and final instruction. In the eight-shot condition, the message began "Here are labeled examples drawn only from the training portion of this fold:" and then repeated the block "Example: [marked demonstration]. Label: [combined label]" eight times before the zero-shot target block. The structured output schema permitted the following eight labels, in order: PROOF; PROOF, CONTRAST; PROOF, EXCEPTION; PROOF-NEG; PROOF-NEG, CONTRAST; META; META, CONTRAST; and META, EXCEPTION.

For each fold and combined label, one demonstration was selected deterministically from the training instances using a SHA-256 ordering over the fold, label, and instance identifier; no demonstration belonged to a held-out split group. The grouped five-fold split was generated deterministically and kept fixed across all experimental conditions, and deterministic per-request seeds were supplied to OpenRouter. The paired group-bootstrap analysis used 20,000 resamples and seed 20260730.

NaturalPRISM: A Natural Language Premise Retrieval Task for Research-level Intermediate Proof States

Harris Proctor[1], Li An Li[1], Austin LaHue[2], Michael Burr[2], Vinita Gangaram Jansari[2], Carlos Toxtli Hernandez[2], Luis David Garcia Puente[1], Benjamin E. Nye[1]
[1]Colorado College [2]Clemson University

Abstract

As AI use progresses in research-level mathematics, automated information retrieval systems for constructing relevant prompts are increasingly important. Existing benchmarks and datasets for identifying which prior results are necessary to prove a statement do not align well with current workflows in which a natural language proof is in progress, but it is unclear what the next step(s) should be.

We introduce NaturalPRISM, a benchmark task and dataset containing 1.7 million pairs of intermediate proof states and the premise invoked in the next step of the proof. All pairs are extracted from natural language research-level mathematics papers published on arXiv, and span all 32 subject categories.

We evaluate the effect of fine-tuning retrieval models for domain-specific vs. domain-agnostic use cases, and find that the combination of sufficient training data with alignment between the training and testing distributions allows smaller specialized models to dramatically outperform larger general models.

The dataset and all associated code is made available online[1].

1 Introduction

Recent years have seen a dramatic increase in mathematicians using large language models (LLMs) to facilitate a wide range of tasks, including finding examples and counter-examples, proof generation and validation, conjecturing, and literature search.

The frontier of tasks that can be effectively supported with LLMs has been steadily increasing in both depth and breadth; in 2021, cutting-edge problem solving datasets (Hendrycks et al., 2021; Cobbe et al., 2021) primarily consisted of grade school-level problems. By 2025, tractable problems had progressed all the way to competition-level undergraduate difficulty (Luong et al., 2025). By 2026, LLMs being credited with significant contributions to research-level mathematics has become a regular occurrence (Tsoukalas et al., 2026).

One of the modern paradigms for LLM use in mathematics is a human-in-the-loop approach in which mathematicians construct prompts that contain the necessary context for their task, evaluate the generated output, and revise the context as necessary. Automating the retrieval of relevant mathematical text that should be included in the context can assist this process and increase its efficacy.

There is a rich history of work that addresses information retrieval (IR) for different types of mathematical text, and more recently there has been increased effort in building Retrieval Augmented Generation (RAG) systems that leverage IR strategies to pull relevant text from corpora in order to improve performance on downstream tasks.

The success of these systems depend on the ability of models to accurately identify what text should be selected for retrieval, which can be evaluated by testing models against a variety of benchmark tasks that range from retrieving the proof for a theorem, to retrieving the answer for a question.

In this work, we introduce the new benchmark task and associated dataset **NaturalPRISM** (Natural language Premise Retrieval for Intermediate States in Mathematics) containing over 1.7 million query/answer pairs drawn from research-level proofs written in natural language, and investigate the extent to which model performance differs across distinct domains of mathematics.

2 Related Work

There exists a growing interest in using LLMs for autonomous and semi-autonomous mathematical discovery (Ahn et al., 2024). One of the primary areas of interest has been assessing and improving their ability to generate novel proofs of existing conjectures (Feng et al., 2026; Duc and Liberti,

[1]https://github.com/bnye-cc/aiming-rag

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 167–176
October 29, 2026 ©2026 Association for Computational Linguistics

2025; Onda et al., 2025). Considerable effort has also gone into processing the natural language used in mathematical writing (Drechsel et al., 2025), as well as proof and conjecture generation (Feng et al., 2026; Tafjord et al., 2021).

As the use cases for workflows leveraging LLMs move towards research-level tasks, it is increasingly important to establish general-purpose retrieval systems capable of supporting them. Currently, mathematical retrieval has adopted many of the practices of modern RAG and has been shown to improve conjecture and proof generation (Yang et al., 2025; Ju et al., 2026). Matlas (Ju et al., 2026) builds a semantic search engine over millions of mathematical statements extracted from peer-reviewed papers and textbooks, rendering each statement self-contained via additional context before embedding it for dense retrieval. MathNet (Alshammari et al., 2026) pairs a large multilingual problem set with a retrieval task and finds that augmentation helps only when the retriever surfaces structurally relevant neighbors, while Re^{2}Math (Lyu et al., 2026) evaluates retrieval from partial research level proofs. The BRIGHT benchmark (Su et al., 2025) shows that establishing relevance can require reasoning rather than semantic matching, a paradigm in which conventional embedding-based retrievers perform poorly.

ReasonIR (Shao et al., 2025) responds by training a retriever on synthetic queries paired with some negative examples, achieving state of the art reasoning retrieval and downstream RAG gains. Recent best practices also focus on how queries and contexts are assembled. The O1 Embedder (Yan et al., 2025) generates intermediate "thoughts" for a query before retrieving, ReasonIR finds that longer rewritten queries act as a form of test-time scaling (Shao et al., 2025), and RAG+ (Wang et al., 2025) retrieves knowledge jointly with aligned application examples so the model is shown not only relevant facts, but also how to apply them.

In the mathematics domain specifically, identifying which documents are relevant to each other can be particularly challenging: judging semantic similarity often requires specialized domain knowledge, and the most important connections are often those that are less obvious (including cross-domain parallels). Work on mathematical information retrieval predates the popularity of modern RAG systems (Zanibbi et al., 2025), with notable early benchmarks such as MathIR (Zanibbi et al., 2016) from the NTCIR project (Aizawa and Kohlhase, 2021) which introduced the task of matching formulas and expressions to text.

More recently, additional math information retrieval benchmark tasks have been introduced, including: matching proofs to useful definitions, lemmas, and theorems (Welleck et al., 2021); identifying missing premises for proofs; and retrieving answers for questions posted to public forums such as the Mathematics Stack Exchange (MSE) (Zanibbi et al., 2020; Mansouri et al., 2021, 2022). A collection of 12 retrieval tasks have been complied into the Mathematical Information Retrieval Benchmark (Haocheng Ju, 2025) and grouped into four top-level categories: semantic statement retrieval, question-answer retrieval, premise retrieval, and formula retrieval.

The premise retrieval task category is focused on finding definitions, theorems, and lemmas that would be useful for augmenting a partially completed proof. This task closely aligns with how a mathematician might use an LLM to help facilitate incremental progress towards finishing a proof, without optimistically relying on the wholesale generation of an end-to-end proof.

Existing premise retrieval benchmarks are often built for formal languages - such as Lean4 (Yang et al., 2023), Light (Kaliszyk et al., 2017), and Isabelle (Mikuła et al., 2024) - due to the natural pairing between a partial proof state and the tactics or dependencies necessary to advance to the final proof state. These tasks are excellent for evaluating mathematical reasoning in models, but do little to illuminate a typical mathematicians use case since so few work in formal languages.

On the other hand, benchmarks for natural language premise retrieval include NaturalProofs (Welleck et al., 2021), but are limited either by their size (typically a couple thousand pairs), or by only using the theorem statement as the query without an accompanying partial proof. To fully evaluate the ability of a model to support a mathematician in incremental proof generation, we need a benchmark containing research-level partial proofs written in natural language, paired with the premise(s) necessary to advance to the next proof step.

In this work we generate a novel benchmark task of 1.7 million such pairs by scraping arXiv papers and identifying structures in the source LaTeX that allow us to automatically extract the query/answer pairs. Furthermore, the scale of this dataset allows us to investigate the effect of fine-tuning open-source models, as well as separate out performance

across different domains of mathematics.

3 Methods

3.1 Data Collection

The base dataset consists of every arXiv math paper published between April 2007 and the end of 2018, yielding a set of 235,314 papers. We identify theorem-like statements by extracting all text spans that occur inside LaTeX environments representing Lemmas (lemma, lem, lemm), Propositions (proposition, prop, propo), Corollaries (corollary, co, coro), Conjectures (conjecture, conj), Claims (claim), Definitions (definition, defn, defi, def), and Theorems (theorem, thm, mainthm, maintheorem).We then extract all proofs in LaTeX proof environment. Additionally, many papers contained proofs that occur outside of a LaTeX proof environment but inside a dedicated section for the proof; we capture these instances by also extracting any text span in a section containing the case-insensitive keyphrase "proof of."

This process produced 3,809,157 distinct theorem-like statements, 1,845,449 proofs, and 108,362 proof sections as raw LaTeX strings before cleaning. In total, 3,102,293 of these strings (54%) contain instances of custom commands that are defined in the preamble of the source document (using \def or \newcommand); to improve their quality as stand-alone retrieval targets, we replace all occurrences of custom commands without positional arguments with the original LaTeX.

3.2 Task and Query/Answer Extraction

To construct query/answer pairs, we first extract all proofs contained within each document. Then, we identify instances that could be decomposed into first a partial proof (the query), then a referenced premise (the answer), then the remainder of the proof (discarded) as shown in Figure 1. This was achieved by searching for places where the proof body contained a `\ref` tag that linked to a theorem-like statement elsewhere in the document. We truncate the proof after the sentence containing the reference in order to produce a partial proof, and the text of the referenced statement is taken to be the target premise.

It is relatively common for the body of a proof to contain a `\ref` tag linking to the text of the statement being proved. We filter these instances by establishing which statement is being proved in each proof section (by using the label associated with the title of the section, or by proximity in the absence of a label), and ignoring all of the references to the statement itself.

The names of the labels being referenced carry clear signal about what the corresponding theorem-like text is: when you see `\ref{pythag_thm}`, it is no great mystery that the correct referent is "$a^2 + b^2 = c^2$." To avoid this leakage, we replace the `\ref` and label with the generic `[TARGET]` string. In cases where there are other non-target references in the proof body, we also avoid leaking clues by replacing them with generic `[REF]` strings.

Since each query is only allowed to have one correct target by construction, we also remove instances where the query contains multiple references, and the grammatical structure makes it ambiguous which one should be considered the target. For example, consider a sentence ending with "`...\ref{a}, \ref{b}, and \ref{c}.`" Designating the referent `a` as the correct target would yield two false negatives, since retrieving either `b` or `c` should be considered equally valid.

Finally, we filter out instances with an insufficient amount of mathematical text: queries under 80 characters, premises under 30 characters, spans that are too dense with `\ref` tags, and spans that contain too little LaTeX math notation.

These rules are used to ensure that any queries in our dataset hold enough context clues to allude to the statement which is its answer, without explicitly revealing that answer. This process applied to all of our LaTeX papers created a query answer training dataset of 1,779,598 pairs. We create domain-specific subsets by splitting the pairs based on the primary publication tag of the paper in which they were published. The size of each domain is reported in Table 2.

3.3 Training

Many math retrieval benchmarks report the performance of a suite of frontier models on their task(s) (Haocheng Ju, 2025; Alshammari et al., 2026; Lyu et al., 2026). While this establishes a baseline for the difficulty of the proposed tasks and the differences in performance characteristics between models, there is less information available about the effect of fine-tuning frontier models on math retrieval tasks. There are many factors that contribute to this, but one prominent issue is the relatively small size of the preexisting query/answer datasets. The size of our dataset allows us to focus our evaluation on the effect of fine-tuning a single

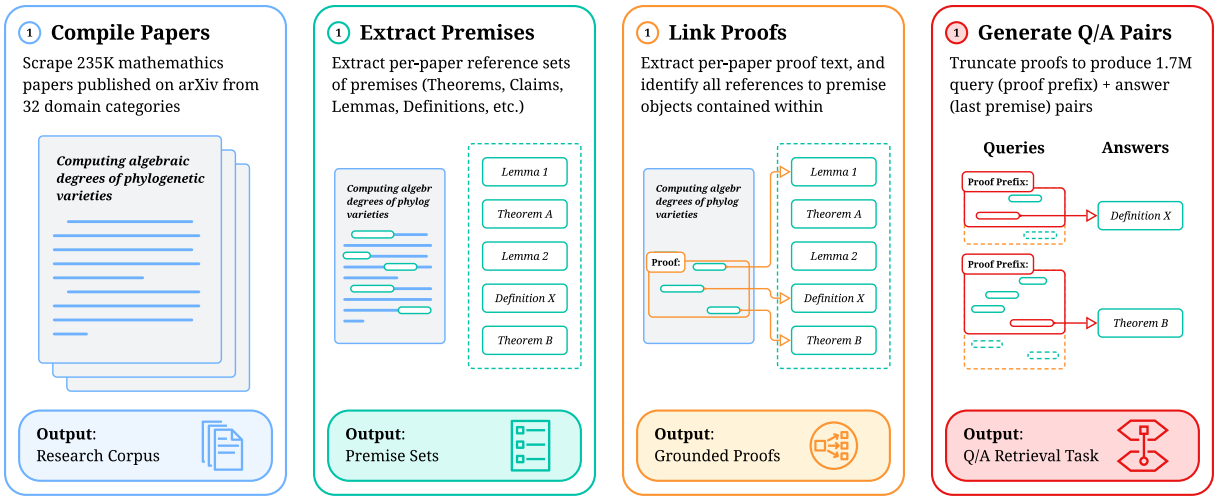


Figure 1: Overview of the Query/Answer pair generation pipeline.

Query	Answer
Suppose that $a(x)$ is a nonconstant polynomial such that every power series coefficient of $a(x)^{-1}$ is 0 or 1. Then [TARGET], $a(x) = (1-x^m)/N(x)$ for some polynomial $N(x)$.	Let $R(x)$ be a rational function with bounded integer power series coefficients. Then $R(x) = N(x)/(1-x^m)$ for some polynomial $N(x)$ and some positive integer m.
Apply [REF] to G with in order to get a set S with $\|S\| \leq (k-1)m$ such that $G \setminus S$ has a partition into components $C_1, \ldots, C_t$ satisfying (i) or (ii). Without loss of generality $\|C_1\| \geq \|C_2\| \geq \cdots \geq \|C_t\|$. If $\|C_2\| + \|C_3\| + \cdots + \|C_t\| \geq m$, then we get $Conn(G \setminus S) = \|C_1\| \leq \|G\| - m$ and so (i) holds. Therefore, we can suppose that $\|C_2\| + \|C_3\| + \cdots + \|C_t\| < m$ holds. In particular this means that $t \leq m-1$, and hence part (b) of [TARGET] holds.	For every k and m, every graph G has a set $S \subseteq V(G)$ with $\|S\| \leq (k-1)m$ such $G \setminus S$ is partitioned into connected componets $C_1, \ldots, C_t$ satisfying one of the following. 1. $t \geq m$. 2. For each $t \in \{1, \ldots, t\}$, C_i is either k-connected or complete.

Table 1: Example query/answer pairs from NaturalPRISM. Queries are a partial proof state (the prefix of a complete proof up until the point of referencing another premise), and the answers are the referenced premise needed to continue the proof.

open-source base model with training sets drawn from different mathematics domains.

We select qwen3-embedding-4B as our base model. This model achieved the best performance on MathNet's Math-Aware Retrieval Task, outperforming several proprietary models (Alshammari et al., 2026). Additionally, this model provides the ability to investigate the effect of model size by comparing against the 0.6B and 8B variants of the same model.

We fine-tune with a contrastive objective including hard negatives and batched negatives. For each query, the model embeds the query and a pool of candidate answers (the true answer, the other statements from the same paper, and the statements from every other paper represented in the training batch) and is trained to rank the true answer above the rest. Queries are formatted with a minimal instruction prompt:

```
Instruct: Given a partial mathematical proof, retrieve the statement it invokes. Query: {query text}
```

We select six different domains to investigate; these collectively span a wide variety of mathematical topics and contain at least 80k query/answer pairs each. The chosen domains are:

For each of the selected domains, we create both a training set and an evaluation set. Additionally,

Category	Papers	QA Pairs	Category	Papers	QA Pairs
math.AC	3,629	57,918	math.IT	4,549	37,158
math.AG	12,655	212,597	math.KT	1,707	29,438
math.AP	13,929	233,850	math.LO	2,983	55,482
math.AT	3,907	60,442	math.MG	2,637	43,186
math.CA	4,970	59,619	math.MP	9,422	131,966
math.CO	14,825	206,453	math.NA	4,652	45,232
math.CT	1,963	40,203	**math.NT**	9,115	133,603
math.CV	3,735	50,494	math.OA	3,472	63,287
math.DG	8,882	126,917	math.OC	6,048	67,058
math.DS	7,479	114,147	**math.PR**	12,123	180,025
math.FA	7,775	113,992	math.QA	3,250	52,303
math.GM	246	2,528	math.RA	4,267	72,512
math.GN	1,285	19,583	math.RT	6,797	126,943
math.GR	5,675	115,260	math.SG	2,133	34,787
math.GT	5,400	81,359	math.SP	2,267	31,022
math.HO	141	1,362	math.ST	3,026	33,538

Table 2: Training pairs and papers per arXiv math category.

arXiv Tag	Category
math.AG	Algebraic Geometry
math.CO	Combinatorics
math.FA	Functional Analysis
math.GT	Geometric Topology
math.NT	Number Theory
math.PR	Probability

we consider two versions of the mixed-domain setting for baseline evaluation sets:

Random All: examples drawn randomly from all areas of math

Balanced All: examples drawn evenly from the six selected categories

Finally, we consider two additional retrieval modes that use different training data:

Random All: fine-tuned on examples drawn randomly from all areas of math

Hydra: a classifier layer first predicts which category of math the query belongs to, and then routes the retrieval task to the appropriate individually trained domain-specific fine-tuned model

4 Results

4.1 Training Dataset Size

We evaluate the retrieval performance of our models using Normalized Discounted Cumulative Gain at 10 (NDCG@10) to assess our models ranking at the top and at broader range. All evaluations

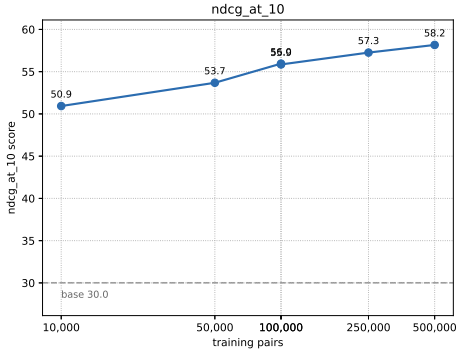


Figure 2: Model performance when trained on different numbers of example Query/Answer pairs. All experiments use the Balanced All training and evaluation sets. The dashed line indicated performance of the base model.

are done without a reranker. Fine-tuning performance of a domain-agnostic qwen3-embedding-0.6b model is shown in Figure 2. We observe that NDCG@10 performance increases steadily with training dataset sizes between 10k and 500k without any evidence of diminishing returns.

To ensure that comparisons of performance across models trained on different categories of math are fair, we must first establish a uniform size for training and evaluation datasets. The Geometric Topology (GT) category contains only 81,359 examples, and as a result we choose to limit all training datasets to 50k examples and all evaluation datasets to 20k examples. We expect that overall performance of the fine-tuned models for both evaluation metrics would increase if we collected additional data or limited ourselves to only the most prolific math categories, but since our primary focus is relative performance between different training strategies we omit these optimizations as future work.

4.2 Domain-Specific Retrieval

Performance of the different training and evaluation settings for a fine-tuned qwen3-embedding-4b model are reported in Table 3. We see that there is an enormous improvement in retrieval performance after fine-tuning across the board: NDCG@10 for the domain-agnostic evaluation (**Random All**) improves from 36.0 to 57.6, and the domain-specific versions show a similar pattern.

Both the training and the evaluation domains have a visible effect on retrieval performance. The mixed domain evaluation settings (**Balanced All** and **Random All**) appear to be easier than the

Model	ARQMath-Task-1	ProofWiki	Stacks	NaturalProofs	NaturalPRISM
Open-source models (≤1.5B)					
gte-large-en-v1.5	37.78	66.49	32.26	28.42	25.34
gte-Qwen2-1.5B-instruct	41.97	77.83	52.56	27.46	-
Qwen3-Embedding (ours)					
qwen3-embedding-4b (base)	47.94	73.41	50.34	33.34	35.95
ALL-4b	44.67	73.15	50.36	34.09	57.62
NT-4b	43.82	70.46	48.10	34.01	58.48
AG-4b	44.45	73.83	51.02	34.48	58.84
CO-4b	47.59	73.06	51.72	33.85	58.10
FA-4b	45.49	72.26	49.04	34.68	59.03
GT-4b	43.89	73.08	47.71	34.21	58.04
PR-4b	46.62	72.72	48.52	35.73	58.24
Hydra-4b	44.78	72.36	50.86	34.50	**59.38**
Open-source models (≥7B)					
e5-mistral-7b-instruct	46.46	77.29	39.85	32.14	33.26
NV-Embed-v2	47.34	83.08	**58.56**	**37.21**	38.23
gte-Qwen2-7B-instruct	44.74	77.02	49.35	30.08	-
SFR-Embedding-2_R	**51.15**	**85.07**	54.94	34.67	39.64
GritLM-7B	46.35	81.59	55.89	32.92	-

Table 3: Comparison with MIRB benchmarks from (Haocheng Ju, 2025). Lower performing models are omitted from the source table for space. Overall best models are in **bold**, and our best performing models are underlined.

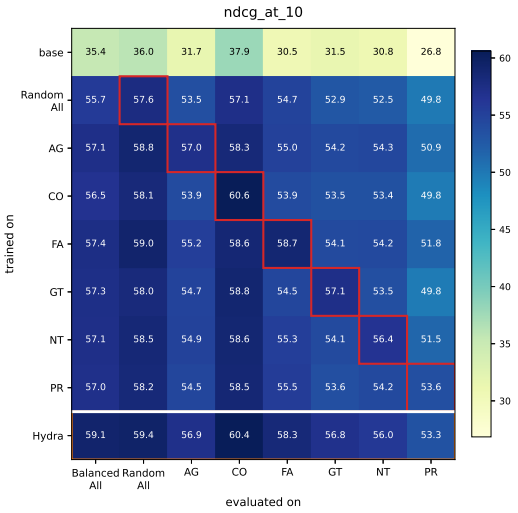


Figure 3: Multiple Fine-tunes of qwen-4B-embedder each fine-tuned on a different area of math, all evaluated on evaluation sets from each domain of math.

domain-specific settings, although this is likely due to the fact that the difficulty of retrieving the correct answer for a given query decreases when many of the candidate answers can be dismissed simply due to a category mismatch.

On the other hand, all fine-tuned models perform significantly better on retrieving Combinatorics answers than on Probability answers - including the model fine-tuned specifically on Probability queries. This demonstrates that there are domain differences in the difficulty of the retrieval task. Performance on domain-specific evaluation sets is always maximized when using the matching training set, offering up to 8% relative improvement over the domain-agnostic training set.

	Predicted						
True	CO	AG	NT	GT	FA	PR	F1
math.CO	**2996**	85	142	73	49	81	0.848
math.AG	101	**3154**	154	102	24	8	0.852
math.NT	196	379	**3013**	64	46	68	0.822
math.GT	161	156	108	**2805**	66	25	0.869
math.FA	111	66	107	65	**3322**	194	0.887
math.PR	74	17	42	29	115	**3189**	0.907

Table 4: Confusion matrix and per-class F1 for the Hydra routing classifier (rows = true category, columns = predicted). Overall accuracy 86.4%, macro-F1 0.864.

Furthermore, the model that uses a classification layer to route queries to the appropriate domain expert (**Hydra**) achieves performance that is only marginally lower than the perfectly domain-aligned settings. As shown in Table 4, this approach achieves 0.864 F1-score on the 6-way classification task; as the size of the training set increases we expect this number to continue to improve, shrinking the performance difference between this model and the domain-aligned settings even further.

Overall fine-tuning significantly improves performance on the retrieval task, and using domain-specific models offers consistent improvement over

the domain-agnostic baseline. We compare results on our benchmark version of the task (**Random All**) against other top-performing retrieval models in Table 3; our **Hydra** model outperforms models almost twice its size by a margin of 50%.

This suggests that when sufficient training data is available for the target information retrieval task, using smaller models than can be fine-tuned with available computational resources is preferable to running inference with generalist frontier models. As reported in Section 4.1, as few as 10k training samples are sufficient to realize a large boost to model performance.

4.3 Benchmark Performance

Task	Input	Target	Domain
ARQMath	MSE Question	MSE Answer	General
ProofWiki	Theorem	Proof	General
Stacks	Theorem	Proof	Algebra, AG
NaturalProofs	Theorem	Theorem-like	Various
NaturalPRISM	Partial proof	Theorem-like	Various

Table 5: Comparison benchmark evaluation tasks from MIRB. *General* indicates that there is no domain structure; *various* indicates that there are separate distinct domains aggregated together.

To evaluate the performance of our math retrieval system on external benchmarks, we choose 4 of the 12 MIRB (Haocheng Ju, 2025) tasks that align most closely with ours: all three Question-Answer Retrieval tasks (ARQMath-Task-1, ProofWiki, and Stacks), and the one natural language Premise Retrieval task (NaturalProofs). We omit the three formal language Premise Retrieval tasks due to the mismatch between languages. A summary of the benchmark tasks is presented in Table 5.

We compute the NDCG@10 for all of our models across the four selected benchmark tasks, and report the results in Table 3. The base qwen3-embedder-4B is already a strong model, outperforming several open source models nearly twice its size. The impacts of fine-tuning are not uniform across benchmarks, and depend on how closely each task resembles our training objective and whether or not there is any alignment between the training and evaluation domains.

For the most dissimilar task, ARQMath, fine-tuning adversely affects performance across the board. For ProofWiki, which involves similar types of text objects but does not have any domain-specific structure, there are no notable improvements from fine-tuning. The Stacks task involves exactly the same text objects as ProofWiki, but is focused specifically on Algebra and Algebraic Geometry problems. We see that our Algebraic Geometry model (and, interestingly, our Combinatorics model) outperform the baseline model.

On NaturalProofs, whose reference retrieval format is closest to our proof and statement training pairs, every fine-tuned model improves over the base model, and our PR fine-tuned model reaches 35.73, second in the table behind only the much larger NV-Embed-v2.

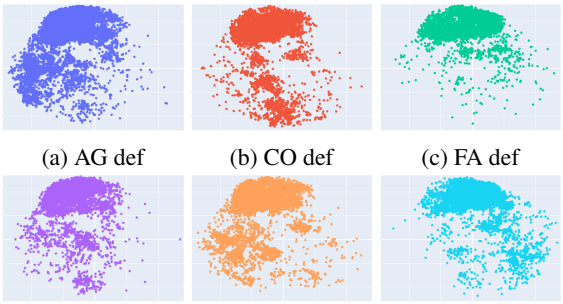
Model	ProofWiki	Stacks	Real Analysis	Number Theory
base	**21.27**	34.71	35.66	41.70
ALL	18.62	37.28	36.08	44.38
AG	18.53	**37.41**	34.54	47.45
CO	18.93	37.03	33.88	45.53
FA	18.72	36.87	37.08	46.07
GT	19.96	37.25	35.11	44.50
NT	18.61	37.24	35.13	45.07
PR	19.64	37.02	**37.30**	**48.97**
Hydra	18.66	37.39	36.89	45.07

Table 6: nDCG@10 on the four NaturalProofs premise-retrieval subsets of MIRB. The MIRB NaturalProofs score is the average of the four. Best single-model score per column in bold.

To further probe the effect of domain-alignment, in Table 6 we break down performance on NaturalProofs across the four different source datasets that it aggregates. Fine-tuning still offers no advantage for the general-domain ProofWiki subtask, but domain-aligned models offer improvement for the Stacks (Algebraic Geometry) and Real Analysis (Functional Analysis) subtasks. For the Number Theory subtask, the domain-aligned model improves over the base model and domain-agnostic fine-tune, but is significantly outperformed by the Probability and Algebraic Geometry models.

Overall, we find that the benefits of fine-tuning drop off as the alignment between tasks decreases. However, when the tasks are similar enough (NaturalProofs) or there is enough domain-specificity to take advantage of (Stacks), fine-tuning on domain-aligned training sets outperforms the base model.

4.4 Theorem Embeddings



(a) AG def (b) CO def (c) FA def

(d) GT def (e) NT def (f) PR def

Figure 4: t-SNE projection of the embeddings of extracted definitions, separated by the domain of the source paper.

As a byproduct of this work, we also visualize the two-dimensional projections of the embedded queries and answers. This allows us to visualize some of the latent structure captured by the embedding model; in Figure 4, we see that there is a common cluster of definitions shared between domains, but smaller domain-specific clusters appear with varying proximity to each other.

We present an interactive browser that allows the user to explore and inspect the embedded objects to investigate the spatial relations between objects. The tool is available via the repository at https://github.com/bnye-cc/aiming-rag.

5 Conclusion

We presented a pipeline for constructing large-scale mathematical retrieval training data directly from the internal reference structure of arXiv papers. By pairing partial proofs with the statements they cite, we assembled a dataset of query answer pairs spanning 235,314 papers, requiring no synthetic query generation or manual annotation. This dataset provides a new benchmark task for evaluating the retrieval quality of models in a setting that directly aligns with an emerging human-in-the-loop workflow paradigm for how mathematicians can use LLMs to accelerate mathematical discovery.

Using this dataset, we fine-tuned Qwen3 embedding models at the 0.6B and 4B scales, both across mathematics generally and within six individual arXiv subject areas. Our experiments yield several useful findings. First, fine-tuning provides a dramatic boost in performance over the base model, but this improvement does not transfer to related Information Retrieval tasks. Second, domain specialized fine-tuning consistently improves retrieval within the target domain without degrading cross domain performance. Our specialized models matched or exceeded the generalist fine-tune on mixed domain evaluations. Third, routing queries to domain specialists via a lightweight classifier achieved the strongest overall results, ranking first on both mixed domain evaluations and second only to the in domain specialist on every category specific evaluation.

Taken together, these results suggest that the AI mathematics community should treat different text types, and even mathematical subfields, as meaningfully distinct retrieval domains that could benefit from model specialization. Although open-source embedding models offer relatively stable performance across a wider variety of retrieval tasks, they can be outperformed by much smaller models that are fine-tuned on a specific task - and even on specific domains of math within that task.

6 Future Work

One direction that follows naturally would be to push performance of the fine-tuned model(s) on this IR task. Due to the limitations of our available resources, we restricted our experimental training datasets to 50k examples and our base model to the 4B paramater version of the qwen3-embedding model. There is clear evidence in Figure 2 that scaling the size of the training dataset will continue to enhance performance, and moving from the 0.6B model to the 4B model yielded an average relative improvement of 5.7%. Experimenting with a wider variety of higher parameter open-source models would likely provide much better absolute scores.

Another direction would be to explore the relationships between text in different domains of mathematics. In this work we simplify the data to six categories, but our method could easily be extended to cover a broader set of topics with varying amounts of training data support. Furthermore, many exciting mathematical discoveries are made when ideas are connected across different domains; we could filter for cases where the answer to a given query appear to come from another domain, and use these examples to construct explicitly cross-domain retrieval training datasets.

Finally, evaluating the impact of our retrieval models on tasks such as proof generation or conjecturing would better contextualize the value of RAG systems for natural language mathematics.

7 Limitations

The reported findings are only evaluated with respect to a single base model, qwen3-embedding-4b. Robustness of these effects with respect to different model architectures or configurations is undetermined.

The dataset itself is constructed via pattern-matching in the source LaTeX, with manually designed filters to improve the quality of the samples. However, there are instances where unusual or malformed text is extracted and not removed by the filtering steps. The exact percentage is unknown, but clustering the embedded text and manually inspecting outliers has confirmed their presence.

Our answers are extracted as they appear in their source paper without additional context, and they often depend on definitions and notation defined elsewhere in the document. We take surface-level steps to mitigate this issue by unwinding document-specific LaTeX commands, but additional preprocessing strategies that bundle relevant context with theorem-like statements, such as explored by Matlas (Ju et al., 2026), could improve retrieval, particularly across papers with differing notation.

The comparison of external models on our NaturalPRISM task in Table 3 is restricted to models evaluated in the original MIRB paper that were also compatible with the FlagEmbedding evaluation framework. More recent embedding models and alternative retrieval architectures are not represented.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *Preprint*, arXiv:2402.00157.

Akiko Aizawa and Michael Kohlhase. 2021. *Mathematical Information Retrieval*, pages 169–185. Springer Singapore, Singapore.

Shaden Alshammari, Kevin Wen, Abrar Zainal, Mark Hamilton, Navid Safaei, Sultan Albarakati, William T. Freeman, and Antonio Torralba. 2026. Mathnet: a global multimodal benchmark for mathematical reasoning and retrieval. *Preprint*, arXiv:2604.18584.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Jonathan Drechsel, Anja Reusch, and Steffen Herbold. 2025. Mamut: A novel framework for modifying mathematical formulas for the generation of specialized datasets for language model training. *Preprint*, arXiv:2502.20855.

Hieu Le Duc and Leo Liberti. 2025. Mathematics with large language models as provers and verifiers. *Preprint*, arXiv:2510.12829.

Tony Feng, Trieu H. Trinh, Garrett Bingham, Dawsen Hwang, Yuri Chervonyi, Junehyuk Jung, Joonkyung Lee, Carlo Pagano, Sang hyun Kim, Federico Pasqualotto, Sergei Gukov, Jonathan N. Lee, Junsu Kim, Kaiying Hou, Golnaz Ghiasi, Yi Tay, YaGuang Li, Chenkai Kuang, Yuan Liu, and 9 others. 2026. Towards autonomous mathematics research. *Preprint*, arXiv:2602.10177.

Bin Dong Haocheng Ju. 2025. Mirb: Mathematical information retrieval benchmark. *Preprint*, arXiv:2505.15585.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Haocheng Ju, Leheng Chen, Peihao Wu, Bryan Dai, and Bin Dong. 2026. Matlas: A semantic search engine for mathematics. *Preprint*, arXiv:2604.17484.

Cezary Kaliszyk, François Chollet, and Christian Szegedy. 2017. Holstep: A machine learning dataset for higher-order logic theorem proving. *Preprint*, arXiv:1703.00426.

Thang Luong, Dawsen Hwang, Hoang H. Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Clara Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu H. Trinh, Quoc V. Le, and Junehyuk Jung. 2025. Towards robust mathematical reasoning. *Preprint*, arXiv:2511.01846.

Zicheng Lyu, Wenjie Yang, Shengzhong Zhang, and Zengfeng Huang. 2026. Re^{2}math: Benchmarking theorem retrieval in research-level mathematics. *Preprint*, arXiv:2605.09012.

Behrooz Mansouri, Anurag Agarwal, Douglas W. Oard, and Richard Zanibbi. 2022. Advancing Math-Aware Search: The ARQMath-3 Lab at CLEF 2022. In *Advances in Information Retrieval*, pages 408–415, Cham. Springer International Publishing.

Behrooz Mansouri, Richard Zanibbi, Douglas W. Oard, and Anurag Agarwal. 2021. Overview of ARQMath-2 (2021): Second CLEF Lab on Answer Retrieval for Questions on Math. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 215–238, Cham. Springer International Publishing.

Maciej Mikuła, Szymon Tworkowski, Szymon Antoniak, Bartosz Piotrowski, Albert Qiaochu Jiang, Jin Peng Zhou, Christian Szegedy, Łukasz Kuciński, Piotr Miłoś, and Yuhuai Wu. 2024. Magnushammer: A transformer-based approach to premise selection. *Preprint*, arXiv:2303.04488.

Naoto Onda, Kazumi Kasaura, Yuta Oriike, Masaya Taniguchi, Akiyoshi Sannai, and Sho Sonoda. 2025. Leanconjecturer: Automatic generation of mathematical conjectures for theorem proving. *Preprint*, arXiv:2506.22005.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen tau Yih, Pang Wei Koh, and Luke Zettlemoyer. 2025. Reasonir: Training retrievers for reasoning tasks. *Preprint*, arXiv:2504.20595.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O. Arik, Danqi Chen, and Tao Yu. 2025. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. *Preprint*, arXiv:2407.12883.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

George Tsoukalas, Anton Kovsharov, Sergey Shirobokov, Anja Surina, Moritz Firsching, Gergely Bérczi, Francisco J. R. Ruiz, Arun Suggala, Adam Zsolt Wagner, Eric Wieser, Lei Yu, Aja Huang, Miklós Z. Horváth, Andrew Ferrauiolo, Henryk Michalewski, Codrut Grosu, Thomas Hubert, Matej Balog, Pushmeet Kohli, and Swarat Chaudhuri. 2026. Advancing mathematics research with ai-driven formal proof search. *Preprint*, arXiv:2605.22763.

Yu Wang, Shiwan Zhao, Zhihu Wang, Ming Fan, Xicheng Zhang, Yubo Zhang, Zhengfan Wang, Heyuan Huang, and Ting Liu. 2025. Rag+: Enhancing retrieval-augmented generation with application-aware reasoning. *Preprint*, arXiv:2506.11555.

Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. 2021. Naturalproofs: Mathematical theorem proving in natural language. *Preprint*, arXiv:2104.01112.

Ruiran Yan, Zheng Liu, and Defu Lian. 2025. O1 embedder: Let retrievers think before action. *Preprint*, arXiv:2502.07555.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. Leandojo: Theorem proving with retrieval-augmented language models. *Preprint*, arXiv:2306.15626.

Tianbo Yang, Mingqi Yan, Hongyi Zhao, and Tianshuo Yang. 2025. Lemmahead: Rag assisted proof generation using large language models. *Preprint*, arXiv:2501.15797.

Richard Zanibbi, Akiko Aizawa, Michael Kohlhase, Iadh Ounis, Goran Topic, and Kenny Davila. 2016. Ntcir-12 mathir task overview. In *NTCIR Conference on Evaluation of Information Access Technologies*.

Richard Zanibbi, Behrooz Mansouri, and Anurag Agarwal. 2025. Mathematical information retrieval: Search and question answering. *Preprint*, arXiv:2408.11646.

Richard Zanibbi, Douglas W. Oard, Anurag Agarwal, and Behrooz Mansouri. 2020. Overview of ARQMath 2020: CLEF Lab on Answer Retrieval for Questions on Math. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 169–193, Cham. Springer International Publishing.

Improving Natural-Language Combinatorial-Optimization Accuracy in Resource-Constrained Language Models via Formal Abstractions

Shrenil Shaun Sharma
Independent Researcher
San Francisco, CA, USA
shrenil19+research@gmail.com

Avi Sharma
Department of Electrical
Engineering and Computer Sciences
University of California, Berkeley
avi_sharma@berkeley.edu

Abstract

Combinatorial scheduling poses a significant challenge for language models, requiring them to identify feasible solutions within exponentially large search spaces while satisfying complex constraints. This challenge is especially pronounced in resource-constrained settings, where larger language models are impractical and selection is limited to smaller models which often fail to preserve feasibility when scheduling directly from natural language. To address these limitations, we introduce SDDL, a neuro-symbolic framework that translates natural-language scheduling problems into compact, solver-aligned representations of tasks, resources, constraints, and objectives, while delegating low-level modeling and search to a deterministic compiler and external solver. On a 300-instance, multi-family subset of scheduling problems, SDDL improves independently verified feasibility for every resource-constrained model tested. The two strongest SDDL configurations reach 55.3% and 28.3%, up from direct-generation baselines of 23.7% and 1.3% and solver-code baselines of 21.7% and 7.0%, with a 0.0% median optimality gap among feasible schedules. By expressing problem structure rather than generating solutions or solver code, SDDL enables smaller models to approach the strongest evaluated direct- and solver-code configurations, including substantially larger frontier models.

1 Introduction

Many requests posed to modern language models arrive in ordinary language, including problems whose solutions depend on an underlying mathematical structure. In such cases, constraints, objectives, and procedures may be specified only implicitly, requiring the model to infer the mathematical problem being described. For tasks that instantiate optimization problems, this inference is not merely semantic: the model must translate a verbal description into a latent search space of possible solutions and evaluate candidates against implicit feasibility and optimality criteria. This challenge is especially pronounced in resource-constrained settings, where model selection is limited to language models with substantially fewer parameters than larger available alternatives. Generating a solution requires parsing the task description, tracking interacting constraints, reasoning about objectives, and implicitly searching over alternatives within a single autoregressive pass. For smaller models, this process often yields fluent but infeasible solutions.

Neuro-symbolic decomposition may mitigate this limitation by separating language understanding from downstream computation: the language model produces an executable or formal representation, and an external runtime or solver performs the corresponding computation (Gao et al., 2023; Pan et al., 2023). In optimization-specific systems, this formalization may include decision variables, constraints, and objectives (Ramamonjison et al., 2022; Ahmaditeshnizi et al., 2024; Shi et al., 2025). This approach, however, shifts the bottleneck to translation fidelity. A solver optimizes only the formalization it receives, so omitted constraints, misdefined variables, or distorted objectives directly undermine the resulting solution. Combinatorial optimization problems expose this weakness particularly well, where unlike small decision problems or clue-based logic puzzles, instances must represent objectives, resource capacities, temporal relations, precedence constraints, and disjunctive alternatives at scale. We find that direct solver-code generation by resource-constrained models often fails as a formalization strategy, producing programs that may be executable and solver-feasible yet unfaithful to the intended problem. To bridge this gap, we introduce **SDDL (Scheduling Domain Definition Language)**, a domain-specific language that narrows the translation target from open-ended solver code to a small set of solver-aligned scheduling primitives. Rather than requiring models to emit

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 177–193
October 29, 2026 ©2026 Association for Computational Linguistics

low-level solver code, SDDL provides scheduling-native abstractions for tasks, resources, constraints, alternatives, and objectives, where SDDL programs are deterministically compiled into a solver model for execution by an external solver. Across 13 models we compare direct and generic solver-code generation, and evaluate **SDDL** across multiple resource-constrained models. SDDL improves independently verified feasibility for every model tested with it, while reducing the median optimality gap among feasible schedules. Its strongest result ranks among the strongest configurations evaluated overall, while others improve to several times their baselines, demonstrating the effectiveness of SDDL relative to direct and generic solver-code generation, and enabling smaller models to match stronger configurations. Our contributions are:

1. **SDDL (Scheduling Domain Definition Language)** a domain-specific language for executable, objective-bearing combinatorial scheduling formulations that improves the fidelity of resource-constrained LLM formalization relative to direct solver-code generation.

2. An independently verified evaluation of **SDDL** against generation strategies matched in conceptual scope and solver target.

3. **An evaluation showing SDDL enables substantially smaller models to match** or closely approach the performance of the strongest direct- and solver-code configurations evaluated, including those using substantially larger frontier models.

2 Related Work

2.1 Structured Formal Reasoning

Empirical evaluations such as PlanBench document substantial weaknesses in systematic, multi-step planning (Valmeekam et al., 2023), while Kambhampati et al. (2024) argue that LLMs are better incorporated into frameworks that delegate planning to external modules. Performance on constrained-generation tasks also varies substantially across model scales, with smaller, resource-constrained models achieving lower constraint-satisfaction rates than larger models, including under zero-shot prompting (Yao et al., 2024). This gap may be compounded by restrictive output-format requirements, which have been shown to degrade performance on reasoning-heavy tasks (Tam et al., 2024). One such formulation arises in combinatorial scheduling problems, where models must simultaneously recover problem semantics, respect representational conventions, and maintain constraint fidelity across interdependent decisions and temporal relations. Together, these findings suggest that preserving reasoning capacity during inference may depend on reducing what the model must formalize and compute, motivating approaches that delegate execution to external tools.

2.2 Solver-Delegated Constraint Reasoning

Neuro-symbolic frameworks leverage LLMs to parse unstructured text into executable representations, delegating computation to external runtimes to bypass internal arithmetic and logical errors (Gao et al., 2023; Chen et al., 2023). Logic-LM extends this approach to symbolic inference, translating natural-language problems into formal logic and using solver feedback to iteratively repair invalid representations (Pan et al., 2023). These results indicate that LLMs can benefit from constructing executable or symbolic representations while delegating deterministic computation and inference to external tools. NL4Opt (Ramamonjison et al., 2022), OptiMUS (Ahmaditeshnizi et al., 2024), and ConstraintLLM (Shi et al., 2025) apply this paradigm to optimization and constraint programming at different levels of abstraction. However, OptiMUS and ConstraintLLM target highly expressive, general-purpose programming environments, where formalization can require verbose variable declarations, low-level solver API calls, and, often, explicit control flow on top of translating the underlying problem semantics; generality that may come at the cost of reliable generation.

2.3 Solver-Free Generation and Decoding

Alternative approaches evaluate LLMs as direct end-to-end combinatorial solvers (Jiang et al., 2026). In such end-to-end generation contexts, evaluation shows that solution quality and feasibility degrade as instance size and structural complexity increase (Jiang et al., 2026). Separately, structured-decoding approaches constrain the generation process itself: syntax-aware parsing (Yin and Neubig, 2018) and grammar-constrained decoding (Geng et al., 2023) enforce structural validity at the token level. However, hard formatting constraints can degrade underlying reasoning performance (Tam et al., 2024), and a syntactically valid representation may still omit a critical

constraint or transfer an incorrect value. SDDL therefore rejects malformed programs at parse time and verifies emitted schedules against the source instance, ensuring syntactic validity alone is not treated as evidence of correctness.

2.4 LLMs for Combinatorial Scheduling

While the paradigms detailed above address general constraint satisfaction, literature targeting scheduling-native problem structure remains narrow and fragmented. Existing scheduling applications mimic the solver-free paradigm through supervised fine-tuning for job-shop domains (Abgaryan et al., 2024). Such approaches collapse problem interpretation and combinatorial search into a single generation step, making it difficult to attribute errors in an infeasible schedule to faulty constraint translation rather than search failure. Conversely, general-purpose CP systems relevant to scheduling (Michailidis et al., 2024; Shi et al., 2025) inherit the heavy formalization overhead of general solver APIs and are not designed to exploit the recurring structural patterns (precedences, resource capacities, coverage requirements, penalized soft constraints) that span scheduling families. Closest to our setting, Logic.py (Kesseli et al., 2025) formalizes search-based problems through a DSL for constraint solving and is evaluated primarily on logic-grid puzzles. Its evaluated system focuses on finding satisfying assignments, rather than optimizing objectives over feasible schedules.

2.5 Natural-Language Scheduling Benchmarks

Scheduling-specific natural-language benchmarks remain limited. Starjob (Abgaryan et al., 2025) provides a large supervised corpus for end-to-end JSSP scheduling, but verbalizes its instances through fixed templates. R-ConstraintBench (Jain and Wetter, 2025) evaluates RCPSP feasibility under systematically varied constraints using similarly structured, field-like descriptions. NL ⇒ Schedule (Liao et al., 2026) offers fuller natural-language descriptions through semi-synthetic instances constructed from real-world materials across four domains. NLCO (Jiang et al., 2026) covers a broader range of combinatorial optimization families outside scheduling and only provides minimally verbalized instances. As our evaluation emphasizes formalization, we prioritize using a benchmark with unambiguous descriptions and canonical source instances, enabling generated schedules to be verified directly against formal ground truth. SCHEDBench (Sharma and Sharma, 2026) provides the best guarantee for this requirement by construction, where each description is a controlled, constraint-preserving verbalization of a canonical source instance drawn from established scheduling literature; providing a definitive formal ground truth and best-known objective for verifying generated schedules.

JSSP instance, 10 jobs × 5 machines

Rules (excerpt). Steps within an item follow the given order; a location processes at most one step at a time; non-preemptive.

Setup. 10 scenes on 5 crew stations. Each scene is an ordered sequence of steps; each step names a station and a duration.

Scene Franklin has a 5-step sequence. Step 1 is at the *North Sound Stage* for 12 hours; Step 2 is at the *Second Unit Screening Room* for 94 hours. . . *[+ 9 more scenes]*

Response format: `<ItemName> step k: start=<int>`, one per line.

Figure 1: Example scheduling problem rendered in natural-language; model must recover the underlying structure and emit a start time per step.

3 Methods and DSL Creation

SDDL's design is based on an observation that automated formalization tends to break not only in understanding the problem, but in the many decisions required to render that understanding effectively (Shi et al., 2025). SDDL removes the failure-prone modeling decisions otherwise left to the model by naming only recurring scheduling structures for the model to identify, while a deterministic compiler handles their downstream encoding. We articulate first principles underlying this stance, then develop the interface that realizes them and the compilation guarantees it provides.

3.1 Scheduling Problems

Scheduling problems require allocating activities to specific time and resource assignments to produce a feasible schedule that satisfies a set of constraints and optimizes an objective. Although they vary widely in form, their constraint structures are drawn from a few recurring relations: precedence between activities, disjunctive resources that process one activity at a time, cumulative resources that admit concurrency within a capacity, and multiple possible execution modes for each activity. Different problem families combine these differently: job-shop scheduling (JSSP) chains operations by prece-

dence over disjunctive machines, whereas resource-constrained project scheduling (RCPSP) replaces machines with cumulative resources, and its multi-mode extension adds mode selection under non-renewable budgets; all NP-hard, with makespan as the standard objective. Instances are constraint-dense (ex. Fig. 1), so a single omitted or misread relation silently changes the feasible region, making scheduling a natural and demanding target for studying faithful formalization of natural-language problem statements.

3.2 Design Principles

We optimize the language for three properties: robustness, ensuring the modeling surface admits few malformed programs; concision, allowing constraints to be expressed without solver-specific machinery; and sufficient expressiveness for the intended problem families. We further require bounded expressiveness: rather than exposing a general-purpose language in which the model may introduce arbitrary variables or predicates, SDDL fixes a vocabulary of recurring scheduling constructs, precedence, disjunctive and cumulative resources, alternative execution modes, alongside supported objectives. This helps reduce opportunities for formalization errors, where otherwise, each degree of freedom exposed to the model creates another opportunity for errors (variable declarations, domain bounds, etc.) none of which specified by the original problem. By fixing the vocabulary, SDDL transfers these decisions to the compiler, recasting the model's role from synthesizing constraints to recognizing pre-constrained scheduling constructs. These properties stem from a deliberate design decision, as fixed grammatical primitives yield a closed surface language, parse-time rejection, and verified compilation. This also improves solver performance, as CP-SAT's global constraints such as `NoOverlap` and `Cumulative`, propagate more strongly than equivalent Boolean decompositions. A free-form encoding may represent a cumulative resource correctly yet bury it in logic that the solver cannot recognize. Because SDDL names these constructs directly, the compiler can consistently emit the strongest global encoding. This bounded vocabulary both prevents formalization errors and preserves propagation strength. The remaining principles then cover a closed surface that rejects malformed programs at parse time, hides solver implementation details, and provides complete coverage without leaving constructs implicit.

$$
\begin{aligned}
\textit{program} &::= \textit{statement}^* \\
\textit{statement} &::= \texttt{task}(\textit{id}, \textit{props}) \\
&\quad \mid \texttt{resource}(\textit{id}, \textit{props}) \\
&\quad \mid \texttt{before}(\textit{id}, \textit{id}) \\
&\quad \mid \texttt{no\_overlap}(\textit{id}) \\
&\quad \mid \texttt{conflict}(\textit{ids}, \texttt{group}{=}s) \\
&\quad \mid \texttt{not\_at}(\textit{id}, \textit{props}) \\
&\quad \mid \texttt{penalize}(m, \texttt{weight}{=}w, \textit{props}) \\
\textit{props} &::= \epsilon \mid \textit{prop}\ (\texttt{,}\ \textit{prop})^* \\
\textit{ids} &::= \textit{id}\ (\texttt{,}\ \textit{id})^* \\
\textit{prop} &::= \textit{key} = \textit{value} \\
\textit{value} &::= \textit{scalar} \mid \texttt{[}\ \textit{value}, \ldots\ \texttt{]} \mid \texttt{\{}\ \textit{value} : \textit{value}, \ldots\ \texttt{\}}
\end{aligned}
$$

Figure 2: The SDDL grammar: * means "zero or more," ϵ the empty string, and $\mid$ separates alternatives. The atoms are *id*, a quoted identifier; *key*, a property name; *m*, a penalty measure; *w*, an integer weight; *s*, a group name; and *scalar*, a number or string.

3.3 Primitives

SDDL defines programs which consist of flat sequences of primitive scheduling constructs, with neither control flow, nor an expression language beyond literal values. Literals include "lists" and "maps", so values such as the `modes` list and `demands` and `consumes` maps are specified directly rather than constructed through code. The complete grammar is shown in Fig. 2 above.

Two of the seven primitives, `task` and `resource`, declare problem objects; four express hard constraints while `penalize` defines soft objectives. Constraint hardness is encoded structurally through the choice of primitive: a clause is hard unless it appears as a penalty, so the model never signals status through weights or phrasing. We selected the seven primitives to maximize coverage while minimizing the language surface, adding one only when existing constructs could not represent the required scheduling pattern. Two design choices illustrate this principle: precedence, which is expressed by the binary `before` primitive, capturing the pairwise finish-to-start ordering commonly used in these problems, avoiding a general temporal operator that the compiler could not otherwise translate uniformly. Second is shared-membership constraints, such as those between courses in the same curriculum, which are expressed using `conflict` with an optional `group` label. The compiler uses this label both to prohibit concurrent assignments, and to identify group-level objective terms. The remaining hard-constraint primitive, `not_at`, forbids a task from executing during specified absolute times.

```
task(id, **props)          resource(id, **props)
before(a, b)               no_overlap(r)
conflict(t1, ..., group=)  not_at(id, **when)
penalize(measure, weight=w, **params)
```

As an illustration, the clause *"operation A runs on machine m_0 ..."* formalizes to the below primitives,

```
task("a", machine="m0", duration=3)
task("b", machine="m1", duration=2)
before("a", "b")
```

whose lowering schedules the two tasks as intervals on their respective machines, then adds the single constraint $start_b \geq end_a$; with the model stating the operations and ordering, while every solver variable is introduced by the compiler.

3.4 Property Polymorphism

To cover varied task structures with a single declaration form, SDDL assigns `task` a property-dependent representation, with each task's form inferred directly from its properties by the compiler. A task is *continuous* if it specifies a duration or modes and no counts. Continuous tasks lower to start, end, and interval variables. This is especially useful for multi-mode activities, where the model supplies a list of duration-and-resource profiles while the compiler introduces the selection variables, optional intervals, and exactly-one constraint needed to choose a single mode. A second structure, discrete tasks declared by count, lowers to day-period-room variables for timetabling families; while specified and compiler-verified, it currently lies outside our evaluated scope. Because well-formed tasks must satisfy exactly one rule, archetype assignment is therefore deterministic, and the model therefore transcribes properties from the problem statement rather than choosing an internal representation. Property polymorphism therefore keeps the language compact while moving error-prone encoding decisions into the compiler.

3.5 Resource Semantics

Resource handling is where the language most clearly justifies its design as it makes explicit a distinction that natural-language descriptions often obscure but solvers must encode differently. A machine serving one job at a time is disjunctive, so assigned intervals cannot overlap. A shared worker pool is cumulative, so tasks may overlap provided total demand does not exceed capacity. Leaving this distinction implicit is a common source of modeling errors. The DSL resolves this ambiguity at declaration time. Renewable resources define a capacity referenced through a per-task `demands` map; their units are occupied during execution and released at completion, so they compile to cumulative constraints. Nonrenewable resources define a total and use a `consumes` map; their units are permanently expended, and compile to a project-wide linear budget. Disjunctive resources use `no_overlap`. The explicit structure determines the encoding: `demands` indicates cumulative capacity, `consumes` indicates exhaustible supply, and `no_overlap` indicates one-at-a-time use. The model need only classify the resource as capacity-limited, exhaustible, or strictly disjunctive, and the compiler then generates the corresponding constraint and prevents encoding errors. All 3 resource classes are exercised in our evaluation.

3.6 Objectives

Where hard constraints determine feasibility, `penalize` optimizes quality within the feasible region. Each penalty specifies a measure and weight; the compiler scales and sums the measures into a single minimized objective. Measures are selected from a fixed set: completion time, overcapacity, day spread, isolation, and room instability; rather than defined as free-form cost functions. This restriction provides the same safeguards as the closed constraint language. Arbitrary objectives can be syntactically valid yet semantically incorrect because of counting errors or optimization in the wrong direction. Named measures are implemented once by the compiler and reused consistently. The model therefore expresses only the intended preference and its weight, while the compiler realizes the objective correctly.

3.7 Compilation

Compilation is deterministic and requires no further model input. Programs are parsed through the language's abstract-syntax machinery accepting only literals, so generated text is never executed, programs do not introduce side effects, and malformed inputs are rejected. Correctness is defined by an abstract, solver-independent semantics specifying schedules and objective values. A schedule assigns start and end times to continuous tasks, including selected durations for multi-mode tasks, and day-period-room tuples to discrete meetings. Each statement defines a feasibility condition, while `penalize` defines a weighted objec-

tive. For example, `before(a,b)` requires (a) to end before (b) starts, `no_overlap` forbids simultaneous resource use, and a renewable `resource` of capacity (k) limits total active demand to (k). The compiler lowers each statement into solver variables, constraints, and an objective. Correctness requires the compiled model to preserve both feasibility and objective values. Direct primitives map immediately to solver constraints; cumulative resources, mode selection, and penalty measures involving reified constraints require additional equivalence arguments, provided in Appendix A.3. For continuous tasks, the inferred horizon is $H = T + \sum_i \max_{m \in M_i} d_{im}$, where T is one plus the latest forbidden time and M_i is task i's set of modes. Every feasible instance admits a serial schedule within this horizon; therefore, under the evaluated makespan objective, the horizon contains at least one optimal schedule.

3.8 Guarantees and Robustness

A closed vocabulary and deterministic compiler provide two guarantees normally unavailable when each instance is independently translated into solver code: every well-formed program compiles to a unique solver model without a new translation for each problem, and adequacy ensures that compilation preserves its meaning. Any feasible solution can be read as a horizon-bounded schedule satisfying the program, and every such schedule corresponds to a feasible solver solution with the same objective value. As the reference semantics and compiled model are constructed statement by statement, adequacy can be proven for each statement independently. For well-formed programs, adequacy removes compiler lowering as a source of semantic discrepancy, leaving the model-produced formalization as the remaining source of semantic error. The failure surface is therefore minimized, with every schedule tested directly against the source instance by an independent verifier. Extension beyond evaluated families requires no redesign of the language: discrete timetabling reuses the same declaration form through a second task archetype, and family-specific objectives are added by registering named measures in the compiler.

4 Experimental Evaluation

We use publicly released benchmark instances, evaluation harness, and verifiers to evaluate SDDL. We include previously reported DIRECT and SOLVER results from SCHEDBench, with newly generated results marked explicitly for contextual comparisons to characterize performance across model capabilities and generation modalities, while SDDL tests DSL-assisted combinatorial optimization in resource-constrained models. All conditions use fixed, instance-independent zero-shot prompts, greedy decoding (temperature 0), and no worked examples. Transient API failures are retried until generation completes, irrespective of solution quality. DIRECT includes only task and format instructions. SOLVER provides an OR-Tools CP-SAT guide covering equivalent scheduling concepts, conventions, and solver targets as SDDL; SDDL additionally includes its language specification due to its absence from pretraining corpora. SOLVER and SDDL therefore share conceptual scope but differ in formalization interface: open-ended solver code versus a closed language with deterministic compilation. Appendix B.3 gives the full inference settings, sandbox configuration, and pinned software and OR-Tools versions.

4.1 SCHEDBench Evaluation Subset

We evaluate SDDL on the JSSP, single-mode RCPSP, and multi-mode RCPSP families of SCHEDBench, which dominate standard benchmarks and prior work on LLM scheduling. These families exercise the precedence, disjunctive and cumulative resources, nonrenewable budgets, multi-mode selection, and the objective channels of the DSL, and use a standardized makespan objective `penalize`, making optimality gap a clean measure of end-to-end feasibility; conflict, not_at, and the discrete archetype are specified and compiler-verified but left unexercised. We exclude families scored through benchmark-specific weighted soft violations, whose instance specific penalty terms could be supported by registering additional named measures in the existing compiler.

4.2 Evaluation Scoring and Metric Definitions

Our primary metric is feasibility, where an emitted schedule must satisfy every hard constraint of the canonical source instance, as determined by an independent verifier. The pipeline renders a schedule as a structured listing of integer start times, checked directly against the source instance rather than the solver's reported status. Violations are classified as precedence, machine-overlap, or resource-capacity errors. For feasible schedules, the verifier recomputes the objective from the schedule

itself, preventing an incorrectly encoded objective from inflating performance. We then report the gap to the canonical optimum or best-known solution; where *optimal* denotes a feasible schedule with a gap of at most (10^{-9}). Gaps are summarized by the median over feasible runs. Each run receives exactly 1 outcome: *feasible*, *infeasible*, *no-solution*, or *run-fail*. Failures before verification, including unparseable DSL outputs, transpilation errors, rendering failures, solver infeasibility, and timeouts; are recorded as *no-solution* or *run-fail*. Feasibility therefore evaluates the complete path from natural-language input to verified schedule, not just the solver's assessment of its own model. We report 95% Wilson CIs for feasibility rates. The DSL is parsed using a literal-only parser, never executed directly.

4.3 Constraint Solving via SAT-Based Solvers

Constraint programming (CP) represents a combinatorial problem using finite-domain decision variables and constraints over joint assignments. Search is interleaved with propagation, which reduces variable domains until a solution is found, infeasibility is proved, or an objective bound is certified. Modern SAT-based solvers implement this through lazy clause generation: propagators express their inferences as clauses for a conflict-driven SAT engine, combining CP propagation with SAT clause learning and linear relaxations for objective bounds. We target CP-SAT, the SAT-based constraint solver in Google OR-Tools. For scheduling, it provides interval variables linking a task's start, duration, and end, with global constraints such as `NoOverlap` for disjunctive resources and `Cumulative` for shared-capacity resources. Their dedicated propagation methods, including overload checking, edge-finding, and energetic reasoning, prune more effectively than pairwise Boolean decompositions. CP-SAT also supports reified linear constraints and `AllDifferent` for assignment problems. Given an integer objective, CP-SAT proves optimality, returns the best solution found within the time limit, or proves infeasibility. Our compiler targets this interface directly, so SDDL primitives map to the solver constructs with the strongest relevant propagation.

5 Results and Discussion

We compare overall feasibility across direct natural-language scheduling (DIRECT), solver-assisted scheduling (SOLVER), and SDDL-assisted configuration (SDDL) for 13 models on the same 300-instance set. The main table and discussion focus on the two strongest resource-constrained SDDL models, while Appendix C reports additional SDDL results for a broader set of resource-constrained models, all of which show similar feasibility improvements. Pairwise feasibility is tested using two-sided McNemar tests on paired instance-level outcomes, with Holm correction applied jointly across pairwise comparisons. Under DIRECT, GPT-5.5 obtains the highest feasibility at 57.0%. Solver-assisted performance is strongly model-dependent: 10 of 13 models improve and 3 decline, with changes ranging from -10.0% to $+52.0\%$ points. Against this baseline, Qwen 3.5 27B with SDDL reaches 55.3% feasibility, while Devstral Small 2 24B with SDDL reaches 28.3% feasibility, both substantially improved over Direct.

5.1 Effect on Resource-Constrained Models

With SDDL, Qwen3.5-27B reaches 55.3% feasibility, up significantly from 23.7% under DIRECT generation, while Devstral-Small-2-24B similarly improves from 1.3% to 28.3%. For Qwen3.5-27B, generic SOLVER assistance yields only 21.7% feasibility, slightly below DIRECT, whereas SDDL delivers gains of 31.7 and 33.7 percentage points over the two conditions, respectively ($p < 10^{-22}$).[1] As both solver-mediated conditions share solver and conceptual scope, these results indicate that solver access alone is insufficient, and supports SDDL's constrained formalization-and-compilation approach. Run-failure rate also falls from 62.7% under SOLVER to 16.0% with SDDL (a 46.7-point reduction), indicating that the feasibility gain coincides with an increase in solve reliability. Devstral-Small-2-24B benefits similarly from SDDL, with feasibility reaching 28.3%, vs. 7.0% with SOLVER and 1.3% under DIRECT ($p < 10^{-15}$), while run-failure falls from 66.7% to 30.0%.

5.2 Performance Positioning of SDDL

We position Qwen 3.5 27B with SDDL against the strongest results in Table 1. The highest direct result is 57.0% feasibility on the 300 instance SCHEDBench subset, obtained by GPT-5.5, while the highest solver-assisted result is 56.7%, obtained by Claude Opus 4.6. Qwen 3.5 27B with SDDL

[1] SDDL discordant gains/losses: 100/5 and 107/6 (Qwen vs. DIRECT and SOLVER); 82/1 and 67/3 (Devstral).

Model	Solver-Mediated Generation					Direct Generation				Δ Feas.
	N	Feas. (%)	95% CI (%)	R. fail (%)	Med. gap (%)	N	Feas. (%)	95% CI (%)	Med. gap (%)	
claude-opus-4-6	300	56.7‡	[51.0, 62.2]	21.0	0.0	300	4.7	[2.8, 7.7]	18.3	**+52.0**
qwen3.5-27b + SDDL†	300	**55.3**‡	[49.7, 60.9]	16.0	0.0	300	**23.7**	[19.2, 28.8]	395.8	**+31.7**
gpt-5.5 (2026-04-23)	300	53.3‡	[47.7, 58.9]	5.0	0.0	300	57.0	[51.3, 62.5]	36.7	−3.7
gpt-5.4 (2026-03-05)	300	51.7	[46.0, 57.3]	18.3	0.0	300	0.3	[0.1, 1.9]	16.7	**+51.3**
gpt-5.4-mini (2026-03-17)	300	36.7	[31.4, 42.3]	32.3	7.5	300	0.7	[0.2, 2.4]	131.8	**+36.0**
claude-sonnet-4-6	300	35.0‡	[29.8, 40.6]	24.3	0.0	300	8.3	[5.7, 12.0]	19.5	+26.7
qwen/qwen3.5-397b (2026-02-16)	300	33.3	[28.2, 38.8]	12.3	0.0	300	19.3	[15.3, 24.2]	14.6	+14.0
devstral-small-2-24b + SDDL†	300	**28.3**‡	[23.5, 33.7]	30.0	0.0	300	**1.3**‡	[0.5, 3.4]	111.2	+27.0
qwen/qwen3.5-122b (2026-02-24)	300	23.3	[18.9, 28.4]	56.0	0.6	300	7.0	[4.6, 10.5]	7.1	+16.3
qwen/qwen3.5-27b (2026-02-24)	300	**21.7**	[17.4, 26.7]	62.7	2.6	300	**23.7**	[19.2, 28.8]	395.8	−2.0
gemini-3.1-flash-lite	300	17.0	[13.2, 21.7]	56.7	0.0	300	0.3	[0.1, 1.9]	107.1	+16.7
gemini-3-flash-preview	300	12.0‡	[8.8, 16.2]	74.3	0.0	300	22.0	[17.7, 27.0]	24.5	*−10.0*
devstral-small-2-24b (2025-12-09)	300	**7.0**‡	[4.6, 10.5]	66.7	0.0	300	**1.3**‡	[0.5, 3.4]	111.2	+5.7
claude-haiku-4-5 (2025-10-01)	300	2.0	[0.9, 4.3]	37.0	0.0	300	1.0	[0.3, 2.9]	135.3	+1.0
meta-llama-4-maverick-17bX123e	300	1.0	[0.3, 2.9]	92.0	0.0	300	0.3	[0.1, 1.9]	150.0	+0.7

Table 1: Per-model results on *SCHEDBench* subset, ‡ marks new results. Brackets list 95% Wilson CIs on feasibility. Bold figures follow model feasibility through all three conditions, direct → solver-mediated → SDDL, with blue for qwen3.5-27b, orange for devstral-small-2-24b. †Evaluated with SDDL, not a new model.

reaches 55.3%, compared with the generic solver-assisted results of GPT-5.5 (53.3%) and GPT-5.4 (51.7%), and within 1.4 and 1.7 points of the strongest solver-assisted (Claude Opus 4.6, 56.7%) and direct (GPT-5.5, 57.0%) configurations. SDDL closes 94.9% of Qwen's deficit to GPT-5.5 under DIRECT (33.3 → 1.7 %) and 96.0% of its deficit to the strongest solver-assisted configuration (35.0 → 1.4 %). The resulting configuration ranks among the highest feasibility overall and exceeds every generic solver-assisted configuration except Claude Opus 4.6 without increasing the capacity of the 27B model. These findings position our approach as a promising means of improving performance across resource-constrained models; with potential applicability to other resource-constrained settings.

5.3 Optimality of Feasible Schedules

Feasibility establishes whether a schedule satisfies the problem's constraints, but not its quality. Thus, we report the median optimality gap among feasible outputs—the difference between a schedule's objective value and the best-known value for its instance, where 0.0% indicates that the schedule matches the best-known objective. Across all four resource-constrained models evaluated with SDDL, the median gap is 0.0%, while feasibility ranges from 15.0% to 55.3%. This represents broader feasible coverage than direct generation, whose median gaps range from 111.2% to 395.8%, and than generic solver-code generation, which generally obtains low median gaps but solves considerably fewer instances. Qwen3.5-27B with SDDL reaches 55.3% feasibility with a 0.0% median gap, compared with 23.7% feasibility and a 395.8% median gap under DIRECT, and 21.7% feasibility and a 2.6% median gap under SOLVER. Since gaps are computed only over feasible outputs, the median for each condition may reflect a different subset of instances, and should be interpreted alongside feasibility. Still, the consistent 0.0% median gap across SDDL configurations indicates its advantage may extend to objective quality beyond validity.

6 Conclusion

We introduced SDDL, a scheduling-specific language that lets models express problem structure through compact, solver-aligned primitives while delegating low-level modeling and search to a compiler and solver. On 300 SCHEDBench instances spanning multiple scheduling families, SDDL substantially raises feasibility, showing that a DSL can serve as an intermediate representation enabling resource-constrained models to match or close the deficit to the strongest evaluated configurations.

Limitations

We do not evaluate SDDL on frontier-scale models, as our focus lies on seeking methods that improve performance for resource-constrained applications where the use of frontier scale sized models may not be possible. SDDL may also improve frontier models however is left as nat-

ural future work. Although SDDL's primitives are designed to cover a wide variety of scheduling problems, its generalization to combinatorial domains beyond scheduling remains empirically untested, and represents a natural direction for future work. Our evaluation also only covers JSSP/SM-RCPSP/MM-RCPSP scheduling families, with discrete formulations and their respective measures specified, compiler-verified, but with evaluation of their LLM-translation left to future work. Our evaluation measures single-pass, zero-shot formalization under greedy decoding. We do not evaluate iterative repair or self-correction which may recover execution failures at additional inference cost. Because models are accessed via provider APIs, temperature-0 decoding is not bitwise deterministic, and each instance is evaluated as a single draw with uncertainty quantified across instances rather than sampling seed. Our use of resource-constrained concerns only the parameter count of the language-model component and does not imply lower end-to-end compute, latency, memory, or cost for the full solver-assisted pipeline.

Based on preliminary pilot experiments, we restrict our evaluation to models with approximately 20B or more parameters. Models below this range produced substantially lower rates of valid formalizations (feas. $< 5\%$), making a full evaluation prohibitively uninformative under our fixed zero-shot setting. This threshold was selected empirically; therefore, our conclusions are limited to the evaluated model-size range and should not be interpreted as establishing a general minimum model size for SDDL.

References

Henrik Abgaryan, Tristan Cazenave, and Ararat Harutyunyan. 2025. Starjob: Dataset for LLM-driven job shop scheduling. *arXiv*, arXiv:2503.01877.

Henrik Abgaryan, Ararat Harutyunyan, and Tristan Cazenave. 2024. LLMs can schedule. *Preprint*, arXiv:2408.06993.

Ali Ahmaditeshnizi, Wenzhi Gao, and Madeleine Udell. 2024. OptiMUS: Scalable optimization modeling with (MI)LP solvers and large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 577–596. PMLR.

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

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

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10932–10952, Singapore. Association for Computational Linguistics.

Raj Jain and Marc Wetter. 2025. R-ConstraintBench: Evaluating LLMs on NP-complete scheduling. *Preprint*, arXiv:2508.15204.

Xia Jiang, Jing Chen, Cong Zhang, Jie Gao, Chengpeng Hu, Chenhao Zhang, Yaoxin Wu, and Yingqian Zhang. 2026. Reasoning in a combinatorial and constrained world: Benchmarking LLMs on natural-language combinatorial optimization. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 30592–30648, San Diego, California, United States. Association for Computational Linguistics.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B. Murthy. 2024. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 22895–22907. PMLR.

Pascal Kesseli, Peter O'Hearn, and Ricardo S. Cabral. 2025. Logic.py: Bridging the gap between LLMs and constraint solvers. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc.

Wenrui Liao, Weihong Du, Yi Li, Hongru Liang, and Wenqiang Lei. 2026. NL $\rightarrow$ schedule: Evaluate multitask scheduling capability of large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 35620–35640, San Diego, California, United States. Association for Computational Linguistics.

Kostis Michailidis, Dimos Tsouros, and Tias Guns. 2024. Constraint modelling with LLMs using in-context learning. In *30th International Conference on Principles and Practice of Constraint Programming (CP 2024)*, volume 307 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:27, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

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

Rindranirina Ramamonjison, Timothy Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, and Yong Zhang. 2022. NL4Opt competition: Formulating optimization problems based on their natural language descriptions. In *Proceedings of the NeurIPS 2022 Competitions Track*, volume 220 of *Proceedings of Machine Learning Research*, pages 189–203. PMLR.

Shrenil Shaun Sharma and Avi Sharma. 2026. SCHED-Bench: A benchmark for evaluating LLM constraint faithfulness in natural-language combinatorial scheduling. *arXiv*, arXiv:2608.00991.

Weichun Shi, Minghao Liu, Wanting Zhang, Langchen Shi, Fuqi Jia, Feifei Ma, and Jian Zhang. 2025. ConstraintLLM: A neuro-symbolic framework for industrial-level constraint programming. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 15999–16019, Suzhou, China. Association for Computational Linguistics.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? A study on the impact of format restrictions on large language model performance. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1218–1236, Miami, Florida, US. Association for Computational Linguistics.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023. PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Advances in Neural Information Processing Systems*, volume 36, pages 38975–38987. Curran Associates, Inc.

Shunyu Yao, Howard Chen, Austin W. Hanjie, Runzhe Yang, and Karthik Narasimhan. 2024. COLLIE: Systematic construction of constrained text generation tasks. In *The Twelfth International Conference on Learning Representations*.

Pengcheng Yin and Graham Neubig. 2018. TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 7–12, Brussels, Belgium. Association for Computational Linguistics.

A SDDL Semantics and Compilation

A.1 Primitive and Property Reference

Table 5 lists every primitive, accepted property, and well-formedness requirement; Table 2 defines the five registered penalty measures.

Measure	Archetype	Definition (minimized)
`makespan`	continuous	$\max_i \mathrm{end}_i$ over all scheduled tasks
`capacity`	discrete	total enrolment exceeding room capacity, summed over assignments
`spread`	discrete	shortfall below each task's `min_days` distinct meeting days
`isolated`	discrete	count of meetings with no adjacent same-group meeting
`room_stability`	discrete	number of distinct rooms used by a task beyond the first

Table 2: The five registered penalty measures. Each is implemented once in the compiler and reused across programs; `makespan` is the only measure exercised by the evaluated families.

A.2 Compiler-Lowering Summary

Table 6 maps each SDDL construct to the CP-SAT encoding the compiler emits; all solver variables are introduced by the compiler.

A.3 Correctness Arguments

Adequacy is proven statement-by-statement; the constructs whose lowerings introduce auxiliary variables require the following arguments:

- **Cumulative resources.** `AddCumulative` enforces $\sum_{i:\, s_i \le t < e_i} d_{ir} \le k$ at every t, the reference condition. Multi-mode tasks contribute one *optional* interval per mode, present iff its mode Boolean holds; `AddExactlyOne` presents exactly the selected mode's interval and demand, so solutions correspond one-to-one with reference schedules.
- **Mode selection.** `AddExactlyOne`$(b_{i\cdot})$ makes selection total and unique; channelling $b_{im} \Rightarrow d_i = d_{im}$ fixes the master interval's duration, per-mode optional intervals share the task's start, and nonrenewable consumption $\sum_m c_{imr} b_{im}$ equals the selected mode's consumption. Starts, durations, consumptions are preserved in both directions.
- **Reified penalty measures.** Every measure lowers to variables constrained to *equal* the measured quantity (`AddMaxEquality` for makespan; complementary `OnlyEnforceIf` pairs, $b \Leftrightarrow$ condition, for discrete counts), never one-sided bounds a minimizer could

exploit; compiled objectives therefore equal reference objectives on all feasible schedules.

- **Horizon soundness.** With $H = T + \sum_i \max_{m \in M_i} d_{im}$ (T = one plus the latest forbidden time; zero for the evaluated families), the serial schedule in topological order is feasible with span $\leq H$, so truncation to $[0, H]$ never empties the feasible set. The unrestricted optimum is at most the serial span $\leq H$, and any schedule attaining it has every end within $[0, H]$; the compiled optimum equals the reference optimum.
- **Composition.** Lowerings share only the task variables (s_i, d_i, e_i, b_{im}) whose meaning the items above fix, so per-statement adequacy composes to program adequacy; the residual failure surface is the model-produced formalization, measured by the independent verifier (B.4).

A.4 Worked Translation Example

Figure 3 shows one complete translation. CP-SAT returns the optimum, makespan 7 (welding: `j0_o0` $[0, 3)$, `j1_o1` $[4, 7)$; grinding: `j1_o0` $[0, 4)$, `j0_o1` $[4, 6)$; the lower bound from Batch Concord's $4{+}3$ chain), rendered back to schedule lines via `label`/`position`.

A.5 Full SDDL, Transpiler, Compiler

- literal-only `ast`-based parser (∼170 lines; generated text is never executed);
- CP-SAT transpiler (∼560 lines; both archetypes, all five measures);
- renderer mapping solved variables to schedule lines via task labels;
- evaluation harness, per-instance outputs, and scoring records.

B Experimental Details

B.1 SCHEDBench Evaluation Subset

Instances chosen randomly using a fixed subsampling seed 42; instance identifiers are the `source_instance` fields of the benchmark. Gaps are computed against canonical optimum or best-known solution; and breakdown of per family subset composition is below.

B.2 Prompt Templates

Figures 4 and 5 reproduce the three fixed, instance-independent system prompts; the user message is the instance's problem text plus its response-format section, identical across tested conditions. All conditions are evaluated zero-shot, without worked examples.

Family	Source suites	N
JSSP	Taillard, DMU, LA, ORB, ABZ, SWV	100
SM-RCPSP	PSPLIB J30–J120	100
MM-RCPSP	PSPLIB MM J10–J30	100

Table 3: Evaluation-subset composition.

B.2.1 Direct Generation

Figure 4 reproduces the DIRECT system prompt.

B.2.2 Generic Solver-Code Generation

Figure 5 reproduces the SOLVER system prompt.

B.2.3 SDDL Generation

The SDDL system prompt is reproduced beginning on p. 17.

B.3 Model, Inference, and Solver Configuration

Setting	Value
Resource-Constrained models	qwen3.5-27b (2026-02-24), devstral-small-2-24b (2025-12-09), qwen3-coder-30b-a3b (2025-07-31), magistral-small-24b (2025-09-17)
Reproduced	pinned versions of Table 1 (SCHEDBench)
Decoding	greedy (temperature 0), zero-shot, single pass, no tools
Max output tokens	96,000 default; provider ceilings; token-terminated responses retained and scored
Sandbox	no network, fresh directory, 4 GB memory, 300 s wall-clock limit; CP-SAT budget fixed at 240 s in both solver-mediated conditions
Software	Python 3.11.2; OR-Tools 9.15.6755 (CP-SAT, default parameters except `max_time_in_seconds`)

Table 4: Inference, sandbox, and solver configuration.

B.4 Verification and Outcome Accounting

The verifier parses emitted schedule lines and rederives feasibility and the objective directly from the canonical source instance; it shares no code with the DSL parser, transpiler, or CP-SAT, and is the same component that scores DIRECT (whose pipeline involves no compiler). Solver status is never trusted.

- **Run-fail**: no scoreable schedule (unparseable/empty output, parse rejection, transpile or runtime error, sandbox timeout, or no completed generation).
- **No-solution**: program executed; solver reported infeasibility or returned nothing within budget.

- **Infeasible**: schedule produced but violates a hard constraint (classified precedence / machine-overlap / resource-capacity, plus coverage for missing activities).
- **Feasible**: all hard constraints verified; objective recomputed from the schedule.
- **DIRECT accounting**: unparseable → run-fail; parsed-but-violating → infeasible; no-solution cannot occur.
- **Denominator**: N=300 per configuration; instances without a completed generation count as run-fail, so each row's outcomes sum to 100%.
- **Wilson 95% CI**: $\big(\hat{p} + \frac{z^2}{2n} \pm z\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}\big)\big/\big(1 + \frac{z^2}{n}\big)$, $z = 1.96$.
- **Paired tests**: two-sided exact McNemar on instance-level feasibility, $p = \min\big(1, 2\sum_{i \le \min(b,c)} \binom{n}{i} 2^{-n}\big)$, $n = b + c$; Holm correction applied jointly across all reported pairwise comparisons; unscored instances count as not-feasible on both sides (no verdict changes on the jointly-scored subset). Table 7 lists all discordant counts.

C Supplemental Results

C.1 Error Analysis

Representative exemplars (one per dominant code):

- **Dropped edge**: all 32 tasks, 4 resources, and every duration/demand of a 30-activity MM-RCPSP instance transcribed correctly; two `before()` entries omitted.
- **Conflation**: instance contains both "Waste Removal Planning" and "North Waste Removal Planning"; the program declares only `north_waste_removal_planning` yet writes `before("waste_removal_planning", ...)` → `KeyError`.
- **Derailment**: correct DSL for 160 lines, then drift into natural-language commentary; rejected at parse time.
- **Duplicate declarations**: a job-shop program emits its 300 operation declarations twice (600 task() calls), leaving edge semantics attached to shadowed duplicates.

Primitive	Property	Value	Meaning / well-formedness
`task(id, ...)`	`duration`	int ≥ 0	processing time; declares a *continuous* task
	`modes`	list of maps	alternative execution modes, each a map with `duration` and optional `demands`/`consumes`; declares a continuous multi-mode task; mutually exclusive with a top-level `duration`
	`demands`	map {rid: int}	renewable units held while active; keys must name declared `resource()` ids with `capacity`
	`consumes`	map {rid: int}	nonrenewable units expended once; keys must name resources with `total`
	`machine`	string rid	fixed disjunctive-resource assignment; pair with `no_overlap(rid)`
	`job`, `position`	int, int	job index and 0-based operation index (job-shop bookkeeping used by the renderer)
	`label`	string	verbatim display name from the problem text; consumed by the renderer/verifier, not the solver
	`count`	int ≥ 1	number of meetings; declares a *discrete* (timetabling) task
	`min_days`, `students`	int	discrete-archetype spread/enrolment attributes
	`demand`	int ≥ 1	units required while assigned (discrete archetype: enrolment, checked against room `capacity` and scored by the `capacity` measure)
`resource(id, ...)`	`capacity`	int ≥ 1	renewable per-time capacity $\Rightarrow$ cumulative semantics; for the discrete archetype, a room's seat capacity
	`total`	int ≥ 1	nonrenewable project-wide budget $\Rightarrow$ linear budget semantics
	(neither)	—	strictly disjunctive resource; meaningful with `no_overlap`
`before(a, b)`	—	task ids	hard finish-to-start precedence: $\mathrm{end}_a \leq \mathrm{start}_b$; both ids must be declared
`no_overlap(r)`	—	resource id	at most one assigned task active on r at any time
`conflict(...)`	`group`	string (opt.)	listed tasks may not occupy the same time slot; group label also keys group-level objective terms (discrete archetype)
`not_at(id, ...)`	`day`, `period`	int	forbids execution at the given absolute time; on continuous tasks the forbidden times enter the horizon offset T (§3.7)
`penalize(m, ...)`	`weight`	int ≥ 1	soft objective term; m must be a registered measure name (Table 2)

Table 5: Complete primitive and property reference. A well-formed `task` satisfies exactly one archetype rule: it is *continuous* iff it specifies `duration` or `modes` and no `count`, and *discrete* iff it specifies `count`; archetype assignment is therefore deterministic and complete, and a task specifying both (or neither) is rejected as malformed.

SDDL construct	CP-SAT lowering
continuous `task` (fixed duration d)	$s_i, e_i \in [0, H]$ (`NewIntVar`); interval `NewIntervalVar`(s_i, d, e_i)
continuous `task` with `modes` M_i	mode Booleans b_{im} (`NewBoolVar`) with `AddExactlyOne`; duration variable $d_i \in [\min_m d_{im}, \max_m d_{im}]$ channelled by $b_{im} \Rightarrow d_i = d_{im}$ (`OnlyEnforceIf`); a master interval on (s_i, d_i, e_i) plus one `NewOptionalFixedSizeIntervalVar`(s_i, d_{im}, b_{im}) per mode
`before(a, b)`	`model.Add`$(s_b \geq e_a)$
`no_overlap(r)`	`AddNoOverlap` over the intervals of tasks with `machine` $= r$
renewable resource (`capacity` $= k$)	`AddCumulative`(intervals, demands, k); multi-mode tasks contribute their per-mode *optional* intervals with mode-specific demands
nonrenewable resource (`total` $= B$)	linear budget $\sum_i \sum_m c_{im}\, b_{im} \leq B$ (fixed-mode consumption enters as constants)
`conflict(...)`	`AddAllDifferent` over the members' slot variables (discrete archetype)
discrete `task` (`count` c)	c meetings, each with day, period, and room integer variables and channelled slot $= \mathrm{day} \cdot N_P + \mathrm{period}$; room assignment respects seat capacity
`not_at(id, day, period)`	continuous tasks: reified exclusion $e_i \leq t \vee s_i > t$ per forbidden time t; discrete tasks: slot-exclusion constraints on the day/period variables, reified per meeting
`penalize("makespan", w)`	$ms \in [0, H]$ with `AddMaxEquality`$(ms, \{e_i\})$; contributes $w \cdot ms$
objective	`Minimize`$\big(\sum$ penalty terms$\big)$

Table 6: Statement-by-statement lowering into CP-SAT. Discrete-archetype tasks lower to day, period, and room integer variables with a channelled slot variable $\mathrm{slot} = \mathrm{day} \cdot N_P + \mathrm{period}$.

Natural-language statement. *2 jobs, 2 machines. Job "Batch Northview": step 1 on Main Welding Station for 3 min, step 2 on West Grinding Station for 2 min. Job "Batch Concord": step 1 on West Grinding Station for 4 min, step 2 on Main Welding Station for 3 min.*

SDDL program (model output):

```
resource("main_welding_station")
resource("west_grinding_station")
task("j0_o0", label="Batch Northview", job=0, position=0, machine="main_welding_station", duration=3)
task("j0_o1", label="Batch Northview", job=0, position=1, machine="west_grinding_station", duration=2)
task("j1_o0", label="Batch Concord", job=1, position=0, machine="west_grinding_station", duration=4)
task("j1_o1", label="Batch Concord", job=1, position=1, machine="main_welding_station", duration=3)
before("j0_o0", "j0_o1")
before("j1_o0", "j1_o1")
no_overlap("main_welding_station")
no_overlap("west_grinding_station")
penalize("makespan", weight=1)
```

Principal compiled constraints:

```
horizon = 12
starts['j0_o0'] = model.NewIntVar(0, horizon, 'j0_o0_s')
ends['j0_o0'] = model.NewIntVar(0, horizon, 'j0_o0_e')
intervals['j0_o0'] = model.NewIntervalVar(starts['j0_o0'], 3, ends['j0_o0'], 'j0_o0_iv')
... # same three lines per task
model.Add(starts['j0_o1'] >= ends['j0_o0'])
model.Add(starts['j1_o1'] >= ends['j1_o0'])
model.AddNoOverlap([intervals['j0_o0'], intervals['j1_o1']]) # main_welding_station
model.AddNoOverlap([intervals['j0_o1'], intervals['j1_o0']]) # west_grinding_station
_ms = model.NewIntVar(0, horizon, 'makespan')
model.AddMaxEquality(_ms, [ends['j0_o0'], ends['j0_o1'], ends['j1_o0'], ends['j1_o1']])
penalties.append(_ms)
total = model.NewIntVar(0, 99999999, 'total')
model.Add(total == sum(penalties))
model.Minimize(total)
```

Figure 3: Worked end-to-end translation: natural-language instance → SDDL → compiled CP-SAT model.

```
You are completing an automated benchmark. The user message contains a scheduling problem and an exact output format. Begin your
 response with the first schedule line and output only the schedule in that format -- nothing else: no preamble, no explanation,
 no reasoning, no markdown formatting (no **bold**, no `inline code`, no triple-backtick code blocks), no XML tags such as <
schedule> or <answer>, no tool tags, no Python code. Do not call any tools, do not invoke external solvers, and do not generate
code to be executed -- solve the problem yourself using only your own reasoning. Your response is fed directly into a parser;
any extra characters cause the response to be discarded.
```

Figure 4: DIRECT system prompt.

Model	Contrast	b	c	p
qwen3.5-27b	SDDL vs. DIRECT	100	5	5.0×10^{-24}
	SDDL vs. SOLVER	107	6	5.2×10^{-25}
devstral-small-2-24b	SDDL vs. DIRECT	82	1	1.7×10^{-23}
	SDDL vs. SOLVER	67	3	9.7×10^{-17}
qwen3-coder-30b-a3b	SDDL vs. DIRECT	70	1	6.1×10^{-20}
	SDDL vs. SOLVER	61	2	4.4×10^{-16}
magistral-small-24b	SDDL vs. DIRECT	41	0	9.1×10^{-13}
	SDDL vs. SOLVER	42	5	2.5×10^{-8}

Table 7: McNemar discordant counts (b: SDDL-only feasible; c: comparison-only feasible) and two-sided p-values for the within-model contrasts; all remain significant after Holm correction ($p_{\mathrm{adj}} < 10^{-6}$).

```
You are completing an automated benchmark. The user message contains a scheduling problem written in natural language and an
exact output format. Do not solve the problem in your head and do not emit a schedule directly. Instead, write a single self-
contained Python 3 program that solves the problem using ortools.sat.python.cp_model and prints the resulting schedule to
standard output in exactly the output format the user message specifies. The first character of your response must be the first
character of the program (typically "from" or "import"); output nothing else -- no preamble, no explanation, no reasoning, no
markdown formatting, no triple-backtick code blocks, no XML tags, no comments outside the program body.

Every piece of instance data must be transcribed from the user message into the program as Python literals. Do not read from any
 file, do not import a data module, do not make any network request, and do not depend on any environment variable, command-line
 argument, or side channel. The program must run to completion on its own with `python program.py` in a fresh directory with no
network access. Allowed imports are the Python standard library and ortools.sat.python.cp_model -- nothing else.

The program's standard output must be, and only be, schedule lines in exactly the format the user message specifies -- the same
character-level format that the direct-schedule condition uses. Do not print status labels, diagnostics, timings, or objective
values; do not wrap the output in any envelope or code fence. If the solver returns INFEASIBLE or UNKNOWN, print nothing and
exit cleanly. The program's standard output is fed unchanged into the same parser used to score direct-schedule outputs; any
extra characters cause the response to be discarded.

## CP-SAT modeling guide

model = cp_model.CpModel(). Give every operation (job shop) or activity
(project scheduling) one interval: start and end as NewIntVar bounded by
horizon = sum of all durations, linked by
NewIntervalVar(start, duration, end). Milestones are intervals of
duration 0.

## Constraints

- Precedence: for each successor entry, model.Add(start_b >= end_a).
- Job-shop machine (one operation at a time): AddNoOverlap over exactly the
  intervals assigned to that machine.
- Renewable resource pool (capacity limit while active): AddCumulative over
  the demanding intervals with their per-task demands and the pool's
  capacity. NEVER encode a renewable pool with AddNoOverlap.
- Nonrenewable budget (consumed once): a single linear constraint,
  model.Add(sum of consumed amounts <= total).
- Alternative execution modes: one BoolVar per mode with AddExactlyOne;
  each mode contributes NewOptionalIntervalVar carrying that mode's
  duration, its demands to the relevant AddCumulative calls, and its
  consumption to the budget constraint.
- Slot-conflict groups (timetabling): AddAllDifferent over the tasks' slot
  variables; forbidden times: model.Add(slot != t).

## Objective

makespan = NewIntVar(0, horizon); AddMaxEquality(makespan, all ends);
model.Minimize(makespan). Solve with cp_model.CpSolver(); set
solver.parameters.max_time_in_seconds = 240.

## Rules

1. One interval per operation or activity, including start and finish
   milestones.
2. One AddNoOverlap per job-shop machine; AddCumulative for every
   capacity-limited pool.
3. One precedence constraint per successor entry; transcribe all of an
   activity's listed successors before moving to the next. Job-shop steps
   are chained consecutively.
4. An activity with one fixed profile uses a plain interval, not modes.
5. Transcribe every duration, demand, capacity, and total exactly as given.
6. Minimize makespan.
7. Print ONLY schedule lines, with display names exactly as written in the
   problem text.
```

Figure 5: SOLVER system prompt.

Model	Solver-Mediated Generation					Direct Generation				Δ Feas.
	N	Feas. (%)	95% CI (%)	R. fail (%)	Med. gap (%)	N	Feas. (%)	95% CI (%)	Med. gap (%)	
qwen3.5-27b + SDDL	300	**55.3**	[49.7, 60.9]	16.0	0.0	300	**23.7**	[19.2, 28.8]	395.8	**+31.7**
devstral-small-2-24b + SDDL	300	**28.3**	[23.5, 33.7]	30.0	0.0	300	**1.3**	[0.5, 3.4]	111.2	+27.0
qwen3-coder-30b-a3b + SDDL	300	**23.3**	[18.9, 28.4]	28.7	0.0	300	**0.3**	[0.1, 1.9]	357.1	+23.0
qwen/qwen3.5-27b (2026-02-24)	300	**21.7**	[17.4, 26.7]	62.7	2.6	300	**23.7**	[19.2, 28.8]	395.8	−2.0
magistral-small-24b + SDDL	300	**15.0**	[11.4, 19.5]	47.3	0.0	300	**1.3**	[0.5, 3.4]	159.9	+13.7
devstral-small-2-24b (2025-12-09)	300	**7.0**	[4.6, 10.5]	66.7	0.0	300	**1.3**	[0.5, 3.4]	111.2	+5.7
qwen3-coder-30b-a3b (2025-07-31)	300	**3.7**	[2.1, 6.4]	74.3	0.0	300	**0.3**	[0.1, 1.9]	357.1	+3.3
magistral-small-24b (2025-09-17)	300	**2.7**	[1.4, 5.2]	88.0	0.0	300	**1.3**	[0.5, 3.4]	159.9	+1.3

Table 8: Resource-constrained models across all three conditions, direct → solver-mediated → SDDL, in the format of Table 1. Brackets list 95% Wilson CIs on feasibility. Color follows one model across rows, with blue for qwen3.5-27b, orange for devstral-small-2-24b, green for qwen3-coder-30b-a3b, purple for magistral-small-24b.

Cond.	Family	N	Feas. (%)	R. fail (%)	Med. gap (%)
SDDL	JSSP	100	96.0	3.0	0.0
SDDL	SM-RCPSP	100	17.0	44.0	0.0
SDDL	MM-RCPSP	100	53.0	1.0	0.0
SOLVER	JSSP	100	53.0	40.0	5.4
SOLVER	SM-RCPSP	100	7.0	68.0	0.0
SOLVER	MM-RCPSP	100	5.0	80.0	0.0
DIRECT	JSSP	100	52.0	0.0	697.4
DIRECT	SM-RCPSP	100	4.0	0.0	0.0
DIRECT	MM-RCPSP	100	15.0	0.0	0.0

Table 9: qwen3.5-27b by family and condition ($N = 300$ convention; run-fail includes instances without a completed generation).

Failure code	qwen3.5	devstral	q3-coder	magistral	Total
Dropped / garbled precedence edges (tasks complete)	6	4	4	5	19
Referenced-but-undeclared real activity	1	2	0	3	6
Phantom identifier (no problem counterpart)	1	1	0	0	2
Duplicate task declarations	0	1	1	0	2
Coverage collapse / label mismatch	0	1	2	0	3
Primitive-argument misuse	0	0	1	0	1
Emission derailment (prose drift, repetition, syntax slip)	2	0	0	1	3
Coded / sampled	10/10	9/10	8/10	9/10	36/40

Table 10: Hand-coded causes over a stratified sample of SDDL failures (one primary code per instance).

The SDDL system prompt.

```
You are completing an automated benchmark. The user message contains a scheduling problem written in natural language, followed
by a "Response Format" section describing schedule lines. IGNORE that Response Format section entirely -- it describes a
different answer mode and does not apply to you. Do not solve the problem and do not emit a schedule. Instead, translate the
problem into SDDL: a small declarative language that a downstream compiler turns into an exact solver model. The first character
 of your response must be the first character of the DSL; output nothing else -- no preamble, no explanation, no reasoning, no
markdown, no code fences, no XML tags, no comments.

Every piece of instance data must be transcribed from the user message into the DSL as literals. Transcription accuracy is the
whole task: a single wrong duration or machine silently produces a valid-looking but wrong answer. Do not omit any task, any
precedence, or any resource.

## The 7 primitives

resource(id, **props) a machine, renewable resource, or nonrenewable budget
task(id, **props) one operation / activity to schedule
before(a, b) HARD: task a finishes before task b starts
no_overlap(resource_id) HARD: one task at a time on that resource
conflict(t1, ..., group=) HARD: listed tasks cannot share a time slot
not_at(id, day=, period=) HARD: the task cannot run at that time
penalize(measure, weight=1) soft objective; measures: makespan,
                       capacity, spread, isolated, room_stability

## Task properties

- duration=N time length
- label="Exact Name" REQUIRED on every task. See the Labels section below.
- machine="rid" fixed resource assignment; pair with no_overlap(rid)
- job=N, position=N job index and 0-BASED operation index within that job
- demands={"r": n} renewable units used WHILE active
- consumes={"b": n} nonrenewable units used ONCE
- modes=[{"duration": N, "demands": {...}, "consumes": {...}}, ...]
                     alternative execution modes; the solver picks exactly one

## Resource properties

- capacity=N renewable: per-time-period limit
- total=N nonrenewable: project-wide budget

## Labels -- read carefully, results are discarded without them

‘label‘ carries the entity’s display name from the problem text, verbatim:
capitalization, spacing, and punctuation exactly as written.

- Job-shop problems: label is the ITEM / JOB name ONLY, never with a step
  suffix; the 0-based step index goes in position=.
- Project problems: label is the ACTIVITY name, including the named start and
  finish milestones. Every activity gets its own task() with its own label.

## Identifiers

‘id‘ and resource ids are sanitized snake_case, never the display name. Keys of
demands={} / consumes={} MUST exactly match the corresponding resource() id.

## Rules

1. One task() per operation (job shop) or per activity (project scheduling),
   including start and finish milestones.
2. One resource() per machine, renewable resource, or nonrenewable budget.
3. One before() per SUCCESSOR ENTRY. Work through the activities in order; for
   each one, transcribe all of its listed successors before moving to the next.
   An activity listing k successors contributes k before() calls -- do not move
   on with a partial successor list. Job-shop steps are chained consecutively.
4. One no_overlap() per job-shop machine. NEVER use no_overlap for a project
   scheduling renewable resource -- capacity= plus demands={} already gives the
   correct cumulative semantics, and adding it makes the model wrong.
5. A mode dict omits resources it does not use. An activity with one fixed mode
   uses plain duration=/demands=/consumes=, not modes=[].
6. End with one penalize() per stated objective, naming the measure that
   matches the problem’s objective and its weight.
7. Output ONLY the DSL calls.
```

Structural Signatures of Correct Mathematical Reasoning Traces

Maria Marina[1,2] **and Alexander Panchenko**[1,2]
[1]AIRI, [2]Skoltech
Correspondence: {m.marina.scientia}@gmail.com

Abstract

Self-consistency improves reasoning by voting over sampled final answers, but the vote discards how those answers were reached. We ask whether the internal organization of traces provides an interpretable signal that frequency misses. For each candidate answer, we cluster recurring intermediate statements into semantic *checkpoints*, retain checkpoint orders that are consistent across traces, and identify relations that appear both directly and with intermediate checkpoints inserted. These relations form answer-local shortcut graphs. Our score is the number of graph edges beyond a spanning tree in the most cross-linked component, normalized by answer frequency and smoothed over nearby sampling budgets. The intuition is deliberately structural: correct traces may agree on core dependencies while varying in local decomposition, whereas a wrong majority can arise from repeated copies of one failure route. On MATH500, the structural selector is mostly neutral when evidence is scarce, but after sufficient sampling it frequently repairs majority errors, reaching up to a 2-point accuracy gain over majority voting.

1 Introduction

Chain-of-thought prompting exposes intermediate reasoning (Wei et al., 2022), and self-consistency improves reliability by sampling multiple traces and returning the modal answer (Wang et al., 2023). Repeated sampling has consequently become a standard form of test-time scaling (Brown et al., 2024; Snell et al., 2025). However, the standard aggregation step – majority voting — reduces each rich reasoning trace to a single categorical vote, discarding information about how the answer was reached. Ten near-duplicate derivations and ten solutions that recover the same mathematical dependencies through different local routes contribute exactly the same support.

We study the information lost by this compression. Our hypothesis is not simply that correct traces are "more diverse." Useful diversity must coexist with *consistency*: traces should preserve recurrent mathematical states and their relative order, while differing in which intermediate states are made explicit. This distinction motivates our notion of a *shortcut*. Suppose several traces agree that state A must precede state B. One trace may realize the dependency directly, $A \to B$, whereas another makes an intermediate argument explicit, $A \to D \to B$. Both traces preserve the same dependency, but the second expands a step that the first takes as a shortcut. We regard a relation as especially informative when both realizations recur across traces: it is stable enough to survive different derivations, yet flexible enough not to be a literal copy of one route. When several such relations overlap, the answer-local graph becomes cross-linked rather than tree-like.

We therefore seek a selector whose decision can be traced back to this observable organization of reasoning. Figure 1 visualizes the central pattern. In the first two repair cases, the correct structural candidate has more excess connectivity relative to its frequency than the wrong majority despite receiving fewer votes; the final pair shows the corresponding failure mode, where a small change in the shortcut graph makes a wrong answer structurally preferable.

Our contribution is a simple answer-local construction that combines *checkpoint consistency* with *route diversity*. We (i) build semantic checkpoint communities across traces, (ii) retain precedence relations that are stable and survive both direct and expanded realizations, and (iii) score the most redundantly connected component by its excess edges per supporting trace. The resulting choice can be inspected as a graph of the evidence that caused it.

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 194–200
October 29, 2026 ©2026 Association for Computational Linguistics

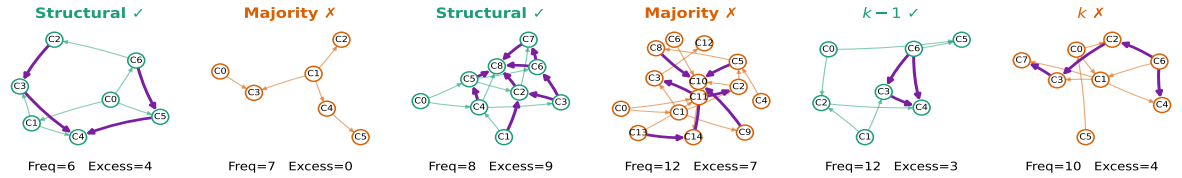


Figure 1: Representative shortcut graphs. The first two pairs are majority-error repairs: the structural selector chooses the correct answer although it is less frequent. In the early case, the structural candidate has frequency 6 and 4 excess edges, while the majority has frequency 7 and zero excess; in the later case the comparison is $8/9$ versus $12/7$ (frequency/excess). The final pair is a structural-drift failure from $k-1$ to k. Thick purple edges show one spanning-tree decomposition of excess connectivity; the number of excess edges is invariant although the particular highlighted edges depend on the chosen spanning tree.

2 Related Work

Self-consistency samples diverse chains but aggregates only final answers (Wang et al., 2023). Universal self-consistency and semantic uncertainty improve answer-level grouping by clustering outputs by meaning (Chen et al., 2023; Kuhn et al., 2023). Learned verifiers and process supervision instead score complete solutions or individual steps (Cobbe et al., 2021; Lightman et al., 2024), while RISC ranks answer candidates using frequency, semantic centrality, and trace-level features (Marina et al., 2026). We focus on a different signal: the *within-answer organization of intermediate reasoning*. Final-answer normalization is fixed, and a fixed semantic matcher exposes recurrent checkpoints whose ordering and connectivity can be inspected directly.

3 Structural Signature of an Answer

3.1 Semantic checkpoints

Intuition. We first look for reasoning steps that play the same semantic role in different traces ending in the *same* answer. These recurring states are our checkpoints. The construction is answer-local because two traces that reach different final answers may reuse similar-looking algebra while supporting different candidate solutions; we therefore never use cross-answer matches to create evidence for a candidate.

For question q, answer a, and set of k sampled traces, let $\mathcal{T}_{q,a}^{(k)}$ contain the traces ending in a, with frequency $n_{q,a}^{(k)}$. We split each trace into ordered reasoning units after removing the final-answer suffix. Every cross-trace pair of units is scored in both directions by Qwen3-Reranker-0.6B (Zhang et al., 2025); the pair score is the minimum directional raw logit. Model's joint pairwise encoding is well suited to fine-grained semantic matching, including mathematical text that mixes natural language and symbolic expressions. Since the score depends on input order, we evaluate both $s(u, v)$ and $s(v, u)$ and use the minimum raw logit as a conservative symmetric similarity score. Within each answer family, we use the empirical 0.90 quantile as the similarity threshold.

We greedily merge mutually similar units using complete-link similarity, while forbidding two units from the same trace in one community. A community is retained only if it occurs in at least 5 distinct traces. Each retained community C_i is a recurring semantic checkpoint. Finally, every original trace is projected back onto its matched communities in trace order. Thus a free-form derivation becomes an ordered checkpoint sequence such as $\pi_t = (C_0, C_2, C_5)$ or $\pi_{t'} = (C_0, C_1, C_2, C_5)$. The next stage compares these discrete sequences rather than the raw text.

3.2 Stable relations with variable local detail

For checkpoints C_i, C_j, let F_{ij} and R_{ij} count traces in which they co-occur in the two possible orders. We require support $F_{ij} + R_{ij} \geq 3$ and define ordering consistency

$$\rho_{ij} = \frac{\max(F_{ij}, R_{ij})}{F_{ij} + R_{ij}}. \tag{1}$$

The larger count determines the dominant direction. For example, if C_0 precedes C_2 in most traces, then the candidate edge is $C_0 \to C_2$.

An edge is inserted in the shortcut graph *only if three conditions hold*: (1) the pair co-occurs in at least 3 traces; (2) its dominant order has consistency $\rho_{ij} \geq 0.80$; and (3) that same order is observed both *adjacently* and *non-adjacently*. Concretely, sequences (C_0, C_2) and (C_0, C_1, C_2) provide the two required realizations of $C_0 \to C_2$: the first uses the relation directly, while the second

inserts checkpoint C_1 between its endpoints. A pair seen only as (C_0, C_2) is repeated but not structurally varied; a pair seen in both orders is varied but not consistent. We keep only the intersection of consistency and local variation. We call the resulting relations *shortcut edges* because one trace can traverse a stable dependency directly while another expands it into several steps.

3.3 Excess connectivity

Let $G_{q,a}^{(k)} = (V, E)$ be the directed shortcut graph for answer a, where V is the set of retained checkpoint communities and E is the set of shortcut edges satisfying the three conditions above. The graph need not be fully connected, so we examine its weakly connected components (WCCs): maximal node sets that are connected after edge directions are temporarily ignored. Direction is used to decide whether a shortcut relation is valid; it is ignored only for the following topological count.

For a WCC H with vertex set V_H and edge set E_H, define

$$\mu(H) = [|E_H| - |V_H| + 1]_+ . \tag{2}$$

For a connected undirected graph, $|V_H| - 1$ edges are sufficient to form a spanning tree. Hence $\mu(H)$ counts exactly how many edges remain beyond that minimum: zero for a tree, one for a component with one independent extra connection, and larger values for increasingly cross-linked structure. We call these *excess edges*. They are the quantity highlighted in purple in Figure 1; the particular spanning tree used for visualization is arbitrary, but the count $\mu(H)$ is invariant.

We keep only the most cross-linked component of an answer and normalize its excess by the number of traces supporting that answer:

$$x_{q,a}^{(k)} = \frac{\max_{H \in \mathrm{WCC}(G_{q,a}^{(k)})} \mu(H)}{\max(n_{q,a}^{(k)}, 1)}. \tag{3}$$

We call this the *largest-component excess-edges-per-trace* score. The normalization matters because a frequent answer has more opportunities to accumulate edges merely by having more traces. At inference time we smooth $x_{q,a}^{(k)}$ with a trailing 5-prefix mean, $\bar{x}_{q,a}^{(k)}$, and choose the answer with the largest smoothed score among candidates observed at least 2 times. Ties are broken by frequency; consequently, if every candidate has zero structural evidence, the procedure reduces to majority voting.

The metric is therefore not a generic connectedness score. Stable checkpoint order supplies *consistency*; requiring both adjacent and non-adjacent realizations establishes that the dependency survives different local decompositions; and excess edges measure how many of those stable shortcuts overlap inside one connected reasoning structure. A repeated failure template can be frequent yet remain tree-like, while correct traces can reconstruct the same dependencies through several compatible routes.

4 Experimental Setup

We use 400 questions from the MATH500 trace corpus training split as a development set for fixing hyperparameters and analyzing their sensitivity (Section 6), while the test split is reserved for final evaluation. MATH500 contains 500 competition-level mathematics problems derived from MATH (Hendrycks et al., 2021; Lightman et al., 2024). We use Meta-Llama-3.1-8B-Instruct (Llama Team, 2024), which generates 99 zero-shot reasoning traces per problem (temperature 0.9, `top_p`=1.0), for 49,500 traces in total. We evaluate nested prefixes $k = 5, \ldots, 99$ against majority voting on the same first k traces. The exact generation and checkpoint-equivalence prompts are provided in Appendix A.

We use a single budget-independent configuration throughout the final evaluation: community support 5, precedence support 3, ordering consistency 0.80, similarity quantile 0.90, minimum answer frequency 2, and rolling window 5. These values are fixed globally rather than tuned separately for individual sampling budgets. Section 6 reports one-factor sensitivity around this reference configuration. We made the code publicly available.[1]

5 Results and Analysis

When does structure help? Figure 2 shows the rolling structural selector across all sampling budgets. With very small prefixes it mostly coincides with majority voting because too few traces support recurrent checkpoint structure. Once enough evidence accumulates, the structural signal becomes useful: the net advantage is positive across most later prefixes, peaks at **10 additional correct questions** (a 2-point accuracy gain on 500 problems),

[1] https://github.com/zlatamaria/structural-signatures

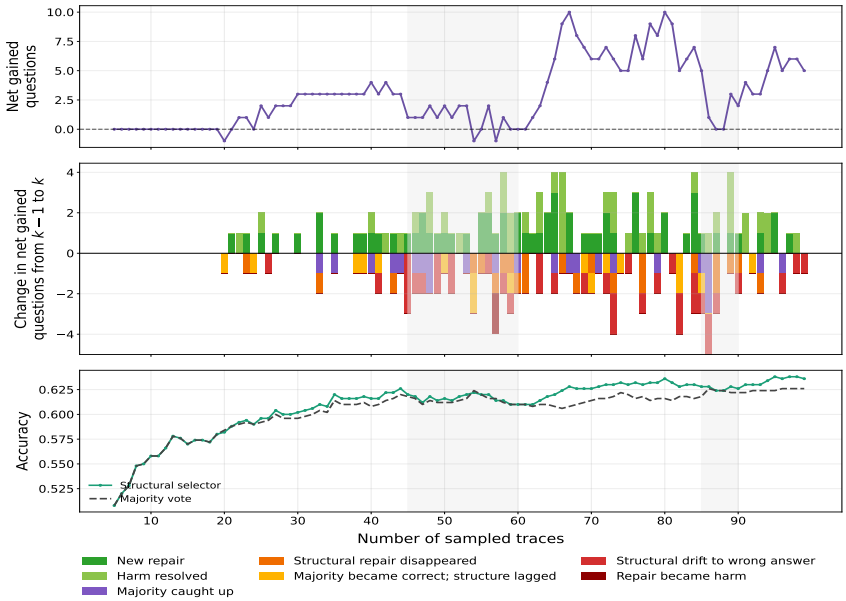


Figure 2: Performance of the rolling structural selector over nested sample sizes. Top: net additional correct questions relative to majority voting. Middle: six question-level transition types that increase or decrease this advantage from one prefix to the next; identical downward changes can reflect baseline catch-up or genuine structural failure. Bottom: matched accuracy of the structural selector and majority. Shaded regions mark the two visible valleys discussed in the text.

and finishes at **+5 questions** at $k = 99$. The bottom panel shows the corresponding accuracy curves: the question-level gains in the upper panels translate into a small but recurrent advantage over majority voting, particularly at larger sample sizes.

Why does the advantage fluctuate? For each question, the structural selector contributes $+1$ relative to majority for a *repair* (majority wrong, structure correct), -1 for a *harm* (majority correct, structure wrong), and 0 otherwise. The middle panel decomposes how this contribution changes from $k-1$ to k into six events. *New repair* is $0 \to +1$ and *harm resolved* is $-1 \to 0$. A lost advantage $+1 \to 0$ has two very different explanations: in *majority caught up*, structure remains correct and majority moves to the same answer; in *structural repair disappeared*, majority remains wrong and structure loses its correction. Likewise, $0 \to -1$ can mean that *majority became correct; structure lagged*, or a genuine *structural drift to wrong answer* while majority stays correct.

The visible valleys are therefore not simply periods in which structural evidence disappears. Around $k = 45$–60, the number of questions with no structural evidence decreases from 227 to 173 and the number of metric ties from 206 to 158, while the mean score margin increases. The dip is driven by a handful of question-level state changes rather than by a global collapse of the graph signal. The late valley is even more illustrative: from $k = 85$ to 86, net gain drops by five questions, but three of the four lost repairs occur because majority changes to the answer that the structural selector was already choosing. In other words, part of the apparent loss is *majority recovery* as additional samples arrive. The remaining new harms are genuine failures in which the shortcut topology changes enough to favor a wrong candidate. This mixture explains why the net-gain curve can be jagged even as aggregate structural evidence becomes denser with larger prefixes.

What does the graph explain? Figure 1 makes both successes and failures inspectable. In the early repair, the wrong majority has 7 traces but zero excess edges, while the correct candidate has 6 traces and four excess edges. In the later repair, majority is again more frequent (12 versus 8), yet the correct candidate has nine excess edges versus seven, giving a larger normalized score ($9/8$ versus $7/12$). The final pair shows the failure mode: the correct choice at $k-1$ has three excess edges over 12 traces, while at k a wrong candidate has four over 10. Thus a difference of only *one extra excess edge*, amplified by normalization, is a really tricky one.

Together, Figures 1 and 2 support *diversity under*

Changed parameter	Reference	Tested value	Mean gain 5–99	Δ vs. ref. 5–99	Mean gain 61–99	Δ vs. ref. 61–99
Reference configuration			**+0.158**	**0.000**	**+0.737**	**+0.000**
Community support	5	2	-2.758	-1.326	-4.026	-1.692
		3	-3.568	-2.137	-4.795	-2.462
Precedence support	3	2	-0.289	-0.447	+0.013	-0.724
		5	-0.261	-0.418	+0.045	-0.692
Ordering consistency	0.80	0.60	-0.255	-0.413	-0.167	-0.904
		0.70	+0.032	-0.126	+0.494	-0.244
		0.90	+0.313	+0.155	+1.038	+0.301
Similarity quantile	0.90	0.85	-0.268	-0.412	-0.527	-1.223
		0.95	-0.220	-0.364	-0.227	-0.923
Rolling window	3	1 (static)	+0.121	-0.037	+0.519	-0.218
		5	+0.150	-0.008	+0.673	-0.064
		7	+0.134	-0.024	+0.853	+0.115
		10	+0.079	-0.079	+0.840	+0.103

Table 1: Hyperparameter sensitivity on the training-derived development set. The reference configuration uses community support 5, precedence support 3, ordering consistency 0.80, similarity quantile 0.90, minimum answer frequency 2, and rolling window 5. Each variant changes only the indicated parameter. Gains are accuracy differences relative to majority voting, in percentage points; Δ vs. ref. is the difference in mean gain from the reference. All variants use the same questions and sampling budgets. Green refers to improvements over selected hyperparameters.

shared structure: recurrent checkpoints preserve a dominant order, while useful answer families realize those dependencies through multiple compatible routes.

6 Ablation

Table 1 reports one-factor sensitivity on the development set: each variant changes only the indicated hyperparameter, while all others remain fixed to the reference configuration. We report both the full range ($k = 5$–99) and $k = 61$–99, since the structural selector provides substantially larger gains over majority voting at higher sampling budgets.

Overall, the fixed configuration is close to a strong operating point: for most hyperparameters, nearby alternatives do not provide consistent improvements. Two exceptions suggest that some parameters are better made budget-dependent. Higher ordering consistency becomes more beneficial at large k, where relations are supported by more traces and stricter agreement can filter occasional reversals without discarding much reliable structure. Longer rolling windows show a similar high-budget advantage. As k grows, the underlying structural signal should change more slowly between consecutive prefixes, while a single added trace can still trigger discrete graph changes by creating or removing edges or components. Stronger smoothing can suppress this local jitter; at small k, however, each new trace carries substantially more information, so a long window may delay genuine changes in the signal. This suggests increasing both the consistency threshold and smoothing horizon with the sampling budget rather than fixing them globally.

7 Conclusion

Majority voting treats traces as interchangeable votes. We instead expose an answer-local reasoning graph and use its excess connectivity as an interpretable selection signal. On MATH500, this signal becomes informative once enough traces are available and can recover correct minority answers that frequency alone misses. Just as importantly, the same graph reveals when the inductive bias fails. This makes structural self-consistency both a reranking heuristic and an interpretable view of what repeated sampling agrees on.

8 Limitations

Checkpoint graphs inherit errors from segmentation and semantic matching. The answer-local quantile threshold is relative rather than an absolute equivalence probability. Cycle rank measures

redundant connectivity, not logical validity: a coherent misconception can also form cycles, as the drift example illustrates. Results are limited to one generator, one mathematics benchmark, and one fixed sequential sample order; alternative models, domains, temperatures, and trace permutations may change the observed regime. Finally, the rolling score introduces temporal dependence across adjacent prefixes, so per-prefix observations should not be treated as independent statistical tests.

References

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

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS 2021*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations, ICLR 2023*. OpenReview.net.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024*. OpenReview.net.

Llama Team. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Maria Marina, Daniil Moskovskiy, Sergey Pletenev, Mikhail Salnikov, Alexander Panchenko, and Viktor Moskvoretskii. 2026. Boosting self-consistency with ranking. *CoRR*, abs/2606.05054. Also published in the ACL 2026 Student Research Workshop.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*. OpenReview.net.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *CoRR*, abs/2506.05176.

A Prompts

A.1 Checkpoint-equivalence instruction

The following instruction is supplied to Qwen3-Reranker-0.6B for each ordered query–document pair; both directions are scored and combined by the minimum raw logit.

```
Judge whether the Document expresses the
same mathematical reasoning checkpoint
as the Query.  Treat paraphrases and
algebraically equivalent formulations
as matches.  The two steps must
express the same intermediate claim,
mathematical transformation, or result.
Reject pairs that merely discuss the
same problem or share vocabulary.
Differences in numerical value,
sign, operator, inequality direction,
condition, assumptions, or conclusion
mean that the checkpoints are not
equivalent.
```

A.2 MATH500 generation prompt

The prompt follows the generation setup of Marina et al. (2026).

System.

```
You are a helpful and expert mathematical
assistant. Solve the given math problem
step-by-step.
```

User.

```
Reason step-by-step to solve the math problem.  At the end of your solution, strictly follow this format to present your final answer:
Final Answer: \boxed{answer}
Ensure the answer within \boxed{} is concise and contains only the final value or expression.
Problem: {problem}
```

Simulating Student-Tutor Interactions with LLMs for Multilingual Math Feedback

Junior Cedric Tonga[1]* **KV Aditya Srivatsa**[2]† **Kaushal Kumar Maurya**[3]†
Fajri Koto[1] **Ekaterina Kochmar**[4]†

[1]Mohamed bin Zayed University of Artificial Intelligence
[2]ETH Zürich [3]IIT Delhi Abu Dhabi [4]University of Victoria

Abstract

Large language models (LLMs) have demonstrated the ability to generate formative feedback and instructional hints in English, making them increasingly relevant for AI-assisted education. However, their ability to provide effective instructional support across different languages, especially for mathematically grounded reasoning tasks, remains largely unexamined. In this work, we present the *first* large-scale simulation of multilingual student-tutor interactions using LLMs. A stronger model plays the role of the tutor, generating feedback in the form of hints, while a weaker model simulates the student. We explore 352 experimental settings across 11 typologically diverse languages, four state-of-the-art LLMs, and multiple prompting strategies to assess whether language-specific feedback leads to measurable learning gains.[1] Our study examines how student input language, tutor feedback language, model choice, and language resource level jointly influence performance. Results show that English hints are generally preferred, even by multilingual student LLMs, with few exceptions that lead to improved student learning outcomes. These findings offer practical insights for developing multilingual, LLM-based educational tools that are both effective and inclusive.

1 Introduction

Large language models (LLMs) have demonstrated strong reasoning abilities in solving mathematical problems, particularly when prompted in English (Kojima et al., 2022; Guo et al., 2025; Bandyopadhyay et al., 2025). A common benchmark for evaluating such capabilities is `GSM8K` (Cobbe et al., 2021), which consists of grade-school-level math word problems. Its multilingual counterpart, `MGSM8K` (Shi et al., 2022), extends this evaluation to a typologically diverse set of languages. Both English-centric and multilingual LLMs perform substantially worse on `MGSM8K` than on its English-only `GSM8K` benchmark (Shi et al., 2022; Ko et al., 2025), highlighting a gap in LLMs' cross-linguistic reasoning abilities. This discrepancy raises questions about the use of LLMs as instructional agents beyond English. Recent work has explored their role as proxy tutors, generating formative feedback and pedagogical hints to support weaker student models or human learners (Wang et al., 2024b; Meyer et al., 2024). One widely studied form of support is hinting: a concise prompt aimed at guiding problem-solving without directly providing the answer. While such interventions have been shown to improve learning outcomes in English (Kochmar et al., 2022), their impact in multilingual settings remains largely unexamined. Moreover, these explorations are conducted in isolation, where either the tutor generates a hint or the student merely consumes it.

In this work, we take a step further by *simulating student-tutor interactions* entirely using LLMs: a stronger model generates hints as a tutor, while a weaker model attempts to solve the problem as a student. This simulation setup allows us to explore the effects of language, hint quality, and prompting strategy in a scalable and reproducible way. Moreover, such LLM-to-LLM simulations can serve as a valuable proxy for real-world educational scenarios, offering insights into how multilingual feedback might impact learning before deploying these systems with actual students. To the best of our knowledge, this is **the first work to simulate multilingual student-tutor interactions between LLMs across a broad range of languages and settings**. Given that effective feedback depends on both linguistic and reasoning proficiency, this raises a central question: *Does multilingual feedback from LLM tutors lead to measurable learning*

*Correspondence to: `junior.tonga@mbzuai.ac.ae`
†Work done while at MBZUAI.
[1]Code on GitHub

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 201–215
October 29, 2026 ©2026 Association for Computational Linguistics

gains in student models?

Our key contributions are as follows:

1. We simulate student-tutor interactions entirely using LLMs, modeling multilingual feedback across 11 languages, multiple prompting strategies, and four LLMs, yielding a large-scale experimental space of 352 settings.
2. We investigate how language-specific feedback influences student performance, examining the interplay between student input language, tutor feedback language, model selection, and whether the language is high- or low-resource, within the domain of mathematically grounded reasoning tasks.
3. We offer practical recommendations for designing LLM-based systems that support effective multilingual feedback and hint generation, highlighting considerations for both research and real-world educational deployment.

2 Related Work

LLMs in Math Reasoning LLMs have demonstrated strong performance in mathematical reasoning tasks, particularly in English, with benchmarks like GSM8K (Cobbe et al., 2021) driving much of this progress. Specialized models such as Minerva (Lewkowycz et al., 2022), trained in scientific texts, achieve state-of-the-art results without relying on external tools. However, these advances are still largely focused on English. Recent benchmarks like MGSM8K (Shi et al., 2022) reveal that multilingual LLMs underperform significantly, especially in low-resource languages, due to limited language coverage and weaker alignment between linguistic and mathematical representations.

Automated Hint Generation Before neural language models, automatic hint generation was typically formulated as a Markov Decision Process (MDP), selecting optimal hints based on student states (Stamper et al., 2008). Subsequent work improved scalability by structuring large hint sets into solution paths, and Paaßen et al. (2017) advanced this by modeling hint policies in continuous edit-distance spaces for better generalization. With LLMs, research has shifted toward direct hint generation: GPT-4 has been paired with a GPT-3.5 "student-validator" to filter poor hints (Phung et al., 2024), and smaller models like LLaMA-3-8B were shown to approach GPT-4o performance when prompts target specific error types (Tonga et al., 2025). Studies show ChatGPT-generated hints can yield learning gains comparable to human hints in mathematics (Pardos and Bhandari, 2024), though quality varies by domain and task complexity, and McNichols et al. (2024) note LLMs still struggle to generalize beyond training data to unseen student errors.

However, much of prior research has focused exclusively on English. This narrow scope limits the applicability of LLM-based tutoring in multilingual learning environments, where students often benefit more from feedback in their native language. Recent work such as MathOctopus (Chen et al., 2024) shows that multilingual tuning can significantly improve math reasoning across languages. However, the generation of effective instructional hints in languages beyond English, especially for low-resource contexts, remains an open challenge.

Simulating student behavior with LLMs. Although persona simulation has long been studied, it has recently gained momentum in LLM-based multi-agent systems for modeling agents with distinct personalities and goal-driven behaviors (Tran et al., 2025). In education, early approaches like SimStudent (Matsuda et al., 2013) aimed for faithful student simulation. Modern LLMs allow for richer, controllable simulations: Benedetto et al. (2024) adjust skill levels when generating exam answers, MathVC creates virtual classrooms with LLM "students" exhibiting diverse misconceptions (Yue et al., 2024), and error-focused datasets like LEMMA help models learn to self-repair (Pan et al., 2025). Recent work, such as Macina et al. (2023), also explores multi-turn dialogues where LLMs act as students in conversations with human tutors. These developments provide ethically appropriate alternatives to real students. *Our work builds on these efforts by treating LLMs as simulated students, aligned with established guidelines and best practices.*

3 Methodology

This section describes our modeling framework, system architecture, and prompting strategies to simulate student-tutor interactions with LLMs. The overall design is illustrated in Figure 1 and consists of the following key components:

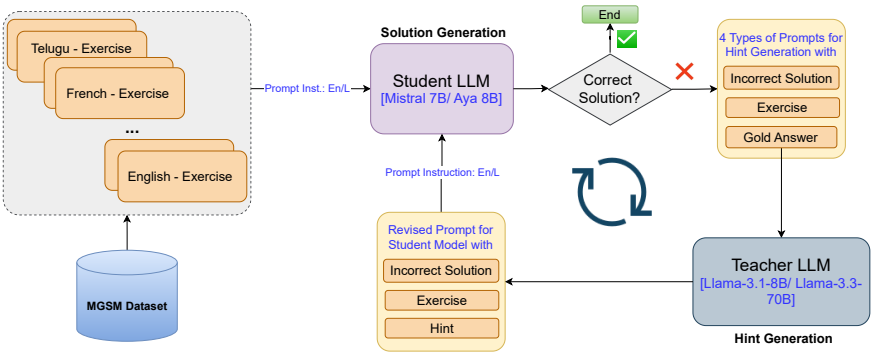


Figure 1: Overview of the student-tutor interaction flow.

Solution Generation The first stage of our pipeline involves generating a candidate student solution to a given multilingual exercise. Let $x \in \mathcal{X}$ represent an exercise in a given language L, and let LLM_S denote the student language model. The model is prompted with x to produce a candidate solution $\hat{y}_S = LLM_S(x)$. To mimic real-world learner behavior, we consider multilingual scenarios where the instructional prompt is given either in English or in the native language of x (i.e., L). This design choice allows us to evaluate the language sensitivity of LLM_S and its downstream performance. We experiment with two student models: the instruction-tuned `Mistral-7B` and the multilingual `Aya-8B`, chosen for their balance between model capacity and efficiency.

Hint Generation For exercises where the generated solution $\hat{y}_S$ is found to be incorrect (i.e., $\hat{y}_S \neq y^*$, where y^* is the reference or gold solution), we employ a tutor model LLM_T to generate pedagogically helpful hints. The motivation here is to simulate intelligent tutoring interventions that guide students toward the correct solution path without directly revealing the answer. The tutor model takes as input a triplet $\langle x, \hat{y}_S, y^* \rangle$ and produces a hint $h = LLM_T(x, \hat{y}_S, y^*)$ under one of four controlled prompting strategies (described in the next paragraph on prompting). GPT-4o (Hurst et al., 2024) was used to validate the correctness of both the initial and revised solutions by comparing them to the gold solution (see prompts in Appendix E.3). We employ large `LLaMA-3.3-70B` LLM as the main tutor but also experiment with the smaller `LLaMA-3.1-8B` LLM.

Prompting We consider two prompting setups for the LLM_S: (1) ***Multilingual prompting***, where the student prompt is written in the same language L as the exercise. To operationalize this, we translate the base prompts (provided in Figure 4 and Figure 5) into the 11 languages of the MGSM benchmark (detailed discussion in Section 4) using Google Translate API;[2] (2) ***English-only prompting***, where the student prompt remains in English, regardless of the exercise's language L. This serves as a control condition to isolate the impact of prompt language on downstream performance.

For hint generation, we explore four strategies by varying the **input prompt language** to the tutor model LLM_T and the **output hint language**:

1. **English-to-English (EN→EN)**: The tutor model (LLM_T) is prompted in English and instructed to generate a hint in English, regardless of the language of the original exercise. This setup uses the prompt format illustrated in Figure 6 of the Appendix.

2. **English-to-English with Translation (EN→EN→L)**: The hint generated in the EN→EN setup is machine-translated with Google Translate into the exercise's native language L. This configuration controls for content while varying the delivery language, enabling analysis of whether presenting hints in the student's native language improves comprehension and learning outcomes.

3. **Native-to-Native (L→L)**: The tutor model is prompted in the native language L of the exercise, using the translated version of the hint prompt provided in Appendix (see Figure 6), and is instructed to generate the hint in the same language L. This aims to explore the impact of native language interaction and hints on the tutor model.

4. **English-to-Native (EN→L)**: The tutor model is prompted in English, following the format shown in Figure 6 of the Appendix, but is explicitly instructed to generate the hint in

[2]Translation performed using Google Translate API from `https://github.com/nidhaloff/deep-translator`.

the target language L. This explores how instruction in English and hinting in the native language affects the tutoring outcomes.

Student-Tutor Interaction Flow The full pipeline of student-tutor interaction is summarized in Algorithm 1. It outlines the interaction between the student and tutor models, including the hint-guided revision loop. Due to computational constraints, we primarily experimented with a single hint iteration ($N = 1$). However, we also include a small-scale analysis for $N > 1$ (see Section **??**), which shows that the key observations made with $N = 1$ largely hold across higher values of N as well.

Algorithm 1 Student-Tutor Interaction Flow

Require: Exercise x in language L, reference solution y^*, maximum hint attempts N
Choose student prompting mode (P_S): *Multilingual* or *English-only*
Generate solution $\hat{y}_S \leftarrow LLM_S(P_S(x))$
if $\hat{y}_S = y^*$ **then**
  return Correct solution
end if
▷ *Hint-Guided Revision Loop*
for $i = 1$ to N **do**
  Choose hint generation prompt strategy P_T
  Generate hint $h \leftarrow LLM_T(P_T(x, \hat{y}_S, y^*))$
  Provide hint h to LLM_S and generate revised solution $\hat{y}_S \leftarrow LLM_S(P_S(x, \hat{y}_S, h))$
  if $\hat{y}_S = y^*$ **then**
    return Correct solution
  end if
end for
return Final student solution $\hat{y}_S$ after N attempts

4 Experimental Setup

Dataset and Languages We use the Multilingual Grade School Math (`MGSM`) dataset (Shi et al., 2022), a multilingual extension of `GSM8K` (Cobbe et al., 2021), consisting of 250 grade-school-level math problems translated from English into 11 typologically and geographically diverse languages (English (en), Bengali (bn), Chinese (zh), French (fr), German (de), Japanese (ja), Russian (ru), Spanish (es), Swahili (sw), Telugu (te), and Thai (th)). These examples are representative of the full `GSM8K` dataset (see Appendix B). Following prior work (Shi et al., 2022), we group the languages into High-Resource Languages (HRLs: en, zh, fr, de, ja, ru, es) and Low-Resource Languages (LRLs: bn, th, te, sw).

Models We use four LLMs: two tutor models, `LLaMA-3.3-70B` and `LLaMA-3.1-8B`, selected for their multilingual strength, and two student models, `Mistral-7B` (monolingual) and `Aya-8B` (multilingual). These cover a range of model sizes and capabilities. Appendix Section C provides solution accuracy scores.

Evaluation Metric We use student gain G as the main evaluation metric, defined as the relative improvement in a student model's accuracy after receiving hints. Let A_{before} and A_{after} denote accuracy before and after hinting (over N iterations). The Student Gain G is:

$$G = \frac{A_{\text{after}} - A_{\text{before}}}{A_{\text{before}}} \times 100 \qquad \bar{G}_{\text{category}} = \frac{1}{L} \sum_{i=1}^{L} G_i$$

This percentage-based metric accounts for the model's initial performance A_{before} and allows for meaningful comparisons (Törnqvist et al., 1985). We average the gain G across all languages within each category (i.e, HRLs and LRLs) to obtain $\bar{G}_{\text{category}}$, with L being the number of languages in the category, and G_i is the gain for language i.

Experimental Space Our setup spans 11 languages, 2 student models, 2 prompt types, 2 tutor models, and 4 hint strategies—yielding **352 unique configurations** (see Table 10 in Appendix). See Appendix Section D for implementation details.

5 Results

In this section, we discuss the impact of different student models and prompts, tutor models and prompts, and variations of student gains across HRLs and LRLs. We present key observations relevant to our core research question on multilingual feedback in mathematical reasoning tasks.

Does the size of the tutor model impact student gains? Based on Appendix Figure A, we observe that `LLaMA-3.3-70B` consistently yields greater gains across both student models and hint generation prompts, outperforming `LLaMA-3.1-8B` for HRLs and LRLs. Specifically, `LLaMA-3.3-70B` achieves higher median gains – for instance, 8.6% in the *multilingual* prompt setup and 10% in the *English-only* with `Mistral` – compared to 7.7% and 8% for `LLaMA-3.1-8B`. This effect is especially pronounced for LRLs, with median improvements reaching up to 31% (`Aya-8B`, *multilingual*) and 38% (`Mistral-7B`, *English-only* setup). While `LLaMA-3.1-8B` also shows strong improvements for LRLs, its performance exhibits higher variability than `LLaMA-3.3-70B`, as evidenced by the presence of outliers and larger interquartile ranges in

		En→En	En→En→L	En→L	L→L
MULTILINGUAL SETUP: *Student input prompt in native language*					
HRLs	Mistral-7B	11.30	10.90	9.50	11.00
	Aya-8B	4.00	3.60	3.60	3.90
LRLs	Mistral-7B	12.10	6.60	5.60	14.90
	Aya-8B	27.60	16.90	17.40	20.50
Avg. HRLs		7.65	7.25	6.55	7.45
Avg. LRLs		19.85	11.75	11.50	17.70
Avg. Overall		13.75	9.50	9.03	12.57
Δ HRLs	Mistral-7B−Aya-8B	7.30	7.30	5.90	7.10
Δ LRLs	Mistral-7B−Aya-8B	-15.50	-10.30	-11.80	-5.60
ENGLISH-ONLY SETUP: *Student input prompt in English*					
HRLs	Mistral-7B	8.70	11.00	12.10	10.60
	Aya-8B	5.90	5.50	4.70	4.70
LRLs	Mistral-7B	46.00	7.00	43.10	8.80
	Aya-8B	35.00	15.10	13.40	15.70
Avg. HRLs		7.30	8.25	8.40	7.65
Avg. LRLs		40.50	11.05	28.25	12.25
Avg. Overall		23.90	9.64	18.33	9.95
Δ HRLs	Mistral-7B−Aya-8B	2.80	5.50	7.40	5.90
Δ LRLs	Mistral-7B−Aya-8B	11.00	-8.10	29.70	-6.90

Table 1: Mean student gains (%) across two experimental setups using LLaMA-3.3-70B as the tutor model. The best average values and second-best values are highlighted. Δ rows indicate performance differences between Mistral-7B and Aya-8B within each language resource category (HRLs and LRLs).

the boxplots, indicating a less equitable distribution for LRLs. *These results suggest that larger tutor models are more effective at generating helpful hints, and that model size plays a key role in mitigating the challenges of low-resource settings.* This is expected as larger models are more capable overall. To simplify the interpretation of our findings, we focus the remainder of our discussion on the large tutor model only.

Does the multilingual student model prompting lead to higher student gains compared to English-only prompting? The *Avg.* labeled rows from Table 1 (distilled from Figure 3) show that *English-only* prompting consistently outperforms *multilingual* prompting across all hint types, except for the L→L setting. This indicates that delivering instructions in English is generally beneficial regardless of hint language. However, when instructions are provided in a native language, the corresponding hints should also be in the native language, especially for LRLs, where this effect is more pronounced. For HRLs, the average gain is comparable across all hint prompts for both setups. Finally, these observations are largely invariant to the type of student LLM, whether monolingual or multilingual. *Overall, multilingual instruction does not necessarily lead to higher student gains unless paired carefully with the hint language.*

Which student model leads to greater gains – multilingual or English-centric? The rows labeled with Δ in Table 1 provide clarity on this question. A negative Δ indicates better performance by the multilingual Aya-8B model, while a positive value favors the monolingual Mistral-7B. For LRLs, Aya-8B tends to be more effective, likely due to stronger language representation. In contrast, Mistral-7B generally performs better on HRLs. Interestingly, the difference in student gains is more pronounced on the average in the *multilingual* setting compared to the *English-only* setting, as indicated by higher absolute values of Δ scores. *Overall, as expected, multilingual LLMs tend to perform better for LRLs, while monolingual LLMs are better suited for HRLs.*

Which hint generation prompt strategy performs best with LLaMA-3.3-70B? Figure 3 and Table 1 present the comparative performance of various hint generation strategies under both *multilingual* and *English-only* settings. The EN→EN strategy yields the highest average improvement across both setups, outperforming other hint prompting strategies. This indicates that models are highly responsive to English prompts. This trend generally holds across both HRLs and LRLs, as well as across different student LLM types. An exception is observed in the *English-only* setting for HRLs when using Mistral-7B as the student LLM, where EN→EN is less effective, possibly due to limited representation of HRLs in Mistral-7B. *Overall, tailoring hint strategies to model- and language-specific characteristics is essential for maximizing student gains, though EN→EN remains the most preferred strategy.*

Does the language of the hint (English vs. the native) influence student gains? Using the larger tutor model, LLaMA-3.3-70B, we find that hint language substantially impacts student model gains as shown in Table 1. In the *multilingual* prompt setup, English hints (generated using the EN→EN tutor prompt) consistently yield the highest median improvements across both student models. Further, for LRLs, Aya-8B achieves its best and most consistent improvement of 31.12%, while Mistral-7B reaches its highest median improvement of 7.6%. In the *English-only* setup, the trend holds: Mistral-7B and Aya-8B obtain their best results for LRLs with English hints, reaching 38.19% and 16.24%, respectively. For HRLs, Aya-8B performs slightly better with translated hints using the

EN→EN→L prompt, whereas `Mistral-7B` benefits more from hints provided directly in the target language (L→L). In contrast, when using the smaller tutor model, `LLaMA-3.1-8B`, no consistent patterns emerge regarding hint language effectiveness. *Overall, English-language hints tend to be more effective when generated by larger tutor models. However, native-language hints can be competitive or even superior in specific cases.*

Student gains across languages. The Avg. row in Table 1 highlights that student models achieve higher gains on LRLs than HRLs in both *multilingual* and *English-only* setups. Multilingual LLMs perform better on LRLs, while monolingual LLMs are more effective for HRLs, consistent with earlier findings. Additionally, among the four hint prompting strategies, the EN→EN prompt yields the highest overall gains across all languages. Further, Table 12 (Appendix) shows that `Mistral-7B` struggles with Telugu and Bengali in the *multilingual* setup but achieves better gains in these languages in the *English-only* setup (EN→EN, EN→L). In contrast, `Aya-8b` demonstrates more consistent and higher gains across LRLs (Telugu, Swahili, Bengali, and Thai) in both setups (see Table 13, Appendix).

Final Takeaways Based on the results and discussion, we summarize our key findings regarding multilingual student–tutor interactions:

1. *Student Prompt:* `English-only` prompts generally perform well; however, when either the hint generation prompt or the hint is in the native language, multilingual prompting may be preferable.
2. *Student Model:* Monolingual models perform better for HRLs, while multilingual models are more effective for LRLs.
3. *Hint Generation Prompt:* EN→EN remains the most preferred strategy, with a few exceptions.
4. *Tutor Model:* Larger models such as `LLaMA-3.3-70B` are generally more effective and should be preferred.
5. *Hint Language:* English hints are generally preferred; however, for HRLs and monolingual student models, native-language hints can be more effective.

		En→En		En→En→L		En→L		L→L	
	Models	**L-3.1**	**L-3.3**	**L-3.1**	**L-3.3**	**L-3.1**	**L-3.3**	**L-3.1**	**L-3.3**
HRLs	`Mistral-7B`	0.51	0.80	0.51	0.80	0.51	0.74	**1.43**	0.63
	`Aya-8B`	0.46	0.51	0.46	0.51	0.17	0.29	**0.57**	0.06
LRLs	`Mistral-7B`	0.70	1.90	0.70	1.90	0.50	1.60	**4.70**	2.20
	`Aya-8B`	1.10	1.80	1.10	1.80	1.20	1.30	**8.30**	1.60

Table 2: Answer leakage proportions (%) across `LLaMa-3.1-8B` (as L-3.1) and `Llama-3.3-70B` (as L-3.3).

6 Further Analyses

Gold Answer Leakage. Despite explicit instructions to the tutor model to avoid revealing the final answer while generating hints, gold answer leakage may still occur due to LLM hallucinations, potentially compromising our findings. To assess this risk, we conduct a *Gold Answer Leakage* test—i.e., *checking whether the final answer appears verbatim within the generated hint.* This is a challenging task, as it requires precise extraction of the gold answer from free-form hints; we adopt a regex-based approach for detection. During hint generation, we flagged any hints that included the gold answer as a *stand-alone number* (i.e., not embedded in a longer number or decimal). Detection used the regex,[3] which matches the exact integer or decimal token when it is delimited by non-digit boundaries and not attached to a decimal point. Table 2 reports the proportion of such hints among the total samples for HRLs (250×7 = 1750) and LRLs (250×4 = 1000), across the four hint generation strategies and student models. The highest observed answer leakage was approximately 8% for `LLaMA-3.3-8B` and around 2% for `LLaMA-3.3-70B`. Since our primary tutor model is `LLaMA-3.3-70B`, this level of leakage is unlikely to significantly impact our findings, especially considering that the regex-based extractors tend to produce some false positives. Notably, the leakage rate is higher for low-resource languages, suggesting greater difficulty in handling those languages.

We further investigated whether the hints that helped students revise their initial answers tended to contain the gold answer. Focusing on the two best-performing strategies in the *multilingual* setup—EN→EN and L→L—we computed the leakage ratio, defined as the number of helpful hints that included the gold answer divided by the total number of helpful hints. These results are presented in Figure 9. The findings suggest that helpful hints rarely reveal the gold answer, with leakage ratios close to zero for most languages. No-

[3] Regex pattern = `r'(?<!\d)' + r'\b' + re.escape(answer_str) + r'\b' + r'(?!\d)'`

		En→En		En→En→L		En→L		L→L	
	Models	Aya-8B	Mistral-7B	Aya-8B	Mistral-7B	Aya-8B	Mistral-7B	Aya-8B	Mistral-7B
HRLs	L-3.1	99.99	99.88	99.88	99.88	99.31	97.62	99.87	99.92
	L-3.3	100.00	99.77	99.94	99.88	99.94	99.71	99.94	99.94
LRLs	L-3.1	99.64	98.90	97.92	**96.90**	98.40	97.85	98.15	98.80
	L-3.3	100.00	99.80	98.95	99.17	99.40	99.37	99.47	99.40

Table 3: Mean language identification accuracy for hints with `LLaMA-3.3-70B` (as L.3.3) and `LLaMA-3.1-8B` (as L.3.1). The lowest number is bold.

		En→En	En→En→L	En→L	L→L
Initial solution					
HRLs	`Mistral-7b`	99.8	99.8	99.8	99.8
	Aya	100.0	100.0	100.0	100.0
LRLs	`Mistral-7b`	96.8	96.8	**96.7**	96.8
	Aya	98.6	98.6	98.6	98.6
Revised solution					
HRLs	`Mistral-7b`	93.1	93.7	94.9	94.3
	Aya	92.0	92.9	93.4	93.0
LRLs	`Mistral-7b`	**90.6**	92.1	92.3	91.8
	Aya	99.4	99.1	98.7	98.9

Table 4: Mean language identification accuracy (in %) of student models' initial and revised solutions. Lowest value is bold for both initial and revised solutions.

table exceptions include Thai and Swahili LRLs, where over 2% of helpful hints contained the gold answer.

Language Consistency of Hints and Student Outputs. *Are the initial solution, generated hints, and revised solution in the intended language?* To verify this, we conducted sanity checks using a FastText-based language identification (LID) method (Bojanowski et al., 2017) with the pre-trained `lid.176` model. Table 5 reports the mean LID accuracy of hints for HRLs and LRLs across hint generation strategies and tutor models. With a minimum LID accuracy of ∼97%, most hints are in the intended language with minor code-mixing. Table 6 presents LID accuracy for initial and revised student solutions. Initial solutions are mostly in the expected language, with a minimum LID accuracy of ∼97%. However, revised solutions show a drop in accuracy to ∼91%, indicating increased code-mixing or language switching. Manual inspection reveals that both tutor models struggle to preserve language consistency in revised solutions, particularly for LRLs—most notably `Mistral-7B`, likely due to its English-centric bias. This degradation may propagate from minor code-mixing in the hints themselves (as previously observed). Interestingly, the EN→L strategy yields the highest LID accuracy in revised outputs, suggesting it is more effective in maintaining language fidelity.

Translation Quality. To evaluate the quality of translated hints in the En→En→L setting—where English hints are translated into target languages using Google Translate—we perform back-translation to English and compute BLEU scores (Papineni et al., 2002) over all samples for each language. As shown in Table 11, translation quality is generally high across languages, with the exception of Telugu, which shows only moderate quality. This further validates the findings with the En→En→L prompt.

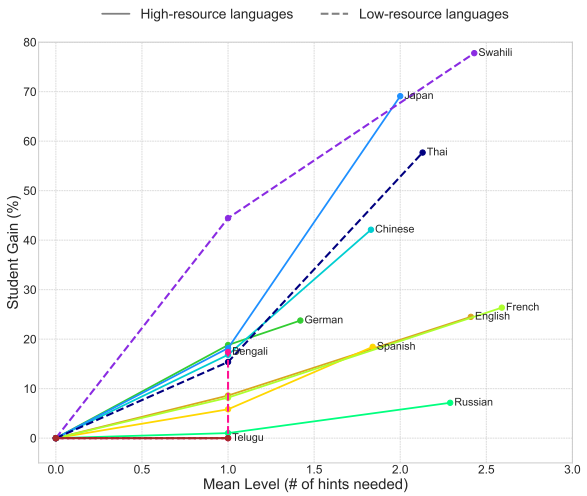


Figure 2: Student Gain scores as a function of the mean number of hints required per language.

		En→En		En→En→L		En→L		L→L	
	Models	Aya-8B	Mistral-7B	Aya-8B	Mistral-7B	Aya-8B	Mistral-7B	Aya-8B	Mistral-7B
HRLs	L-3.1	99.99	99.88	99.88	99.88	99.31	97.62	99.87	99.92
	L-3.3	100.00	99.77	99.94	99.88	99.94	99.71	99.94	99.94
LRLs	L-3.1	99.64	98.90	97.92	**96.90**	98.40	97.85	98.15	98.80
	L-3.3	100.00	99.80	98.95	99.17	99.40	99.37	99.47	99.40

Table 5: Mean language identification accuracy for hints with `LLaMA-3.3-70B` (as L.3.3) and `LLaMA-3.1-8B` (as L.3.1). The lowest number is bold.

Impact of Multiple Hints on Student Gains (N>1). To evaluate whether providing multiple hints enhances student model performance, we consider single setup with L→L strategy—reflecting realistic multilingual tutoring scenarios—using `LLaMA-3.3-70B` as the tutor and `Mistral-7B` as the student. We extend the interaction up to N=5 iterations (i.e., up to five hints), terminating early if the student model produces the correct answer. Figure 2 shows the relative improvement based on the number of hints required. For most languages, performance improves after the first hint, with Swahili showing the largest gain (over 70%). In contrast, Telugu shows no improvement, suggesting that Mistral struggles significantly with this language—likely due to its English-centric training. Interestingly, the average number of hints needed across most languages is around two, indicating that a second hint often contributes meaningfully to student performance. Among high-resource lan-

		En→En	En→En→L	En→L	L→L
Initial solution					
HRLs	Mistral-7b	99.8	99.8	99.8	99.8
	Aya	100.0	100.0	100.0	100.0
LRLs	Mistral-7b	96.8	96.8	**96.7**	96.8
	Aya	98.6	98.6	98.6	98.6
Revised solution					
HRLs	Mistral-7b	93.1	93.7	94.9	94.3
	Aya	92.0	92.9	93.4	93.0
LRLs	Mistral-7b	**90.6**	92.1	92.3	91.8
	Aya	99.4	99.1	98.7	98.9

Table 6: Mean language identification accuracy (in %) of student models' initial and revised solutions. Lowest value is bold for both initial and revised solutions.

guages, German requires the lowest number of hints, with the student typically improving after just one.

7 Conclusion

In this work, we present the first large-scale study involving 352 unique experiments on multilingual student–tutor interactions powered by LLMs, aimed at understanding the effect of language-specific hints across 11 typologically diverse languages, 4 models, and multiple prompting strategies. Our findings reveal that English-centric feedback can enhance student performance, but the most effective configurations vary depending on both the language and the LLM used. Additionally, we observed that even a few iterations of feedback can significantly improve problem solvability. This study offers key insights for designing equitable educational technologies and lays the groundwork for future research in multilingual feedback generation and evaluation. In future work, we aim to extend this research to subject areas beyond mathematics.

Limitations

Although our work advances inclusive, multilingual LLM-based tutoring, several limitations remain and point to fruitful avenues for future research.

LLMs as Tutor–Student Simulators. As in prior studies (Macina et al., 2023; Tran et al., 2025; Wang et al., 2024a), we use LLMs to simulate both the tutor and the student, enabling large-scale, controlled experiments across languages and instructional settings. This strategy yields consistency and broad coverage, but it cannot capture the full diversity of real learners' misconceptions, language proficiency, or problem-solving styles. Likewise, LLM tutors lack the nuanced pedagogical instincts of experienced tutors. Introducing human-in-the-loop evaluations, interactions with real students, and richer student models will increase validity and speed progress toward truly adaptive AI tutors.

Evaluating Hint Quality Beyond Student Gains. We treat a hint as "good" if it leads the student to the correct final answer – that is, if it results in student gains. While practical and easy to measure, correctness alone misses key pedagogical dimensions such as conceptual scaffolding, clarity, and alignment with learning objectives. Expert reviews and rubric-based assessments by mathematics educators could supply these missing perspectives and help refine what counts as a high-quality hint.

Coarse Gain Measurement. Step-level verification of math reasoning with current LLMs is far harder than judging overall solution correctness (Daheim et al., 2024). Consequently, we evaluate feedback with a binary metric – does the student's solution become fully correct or not? This ignores cases where feedback fixes the current error but a later, independent errors may still be present. Developing reliable, fine-grained metrics for partial progress is an important direction for future work.

Lack of Phase-wise Evaluation. Our pipeline follows the two-phase paradigm of first verifying the student's work and then generating a hint (Macina et al., 2023). Ideally, each phase should be evaluated separately. Yet automated assessment of both verification quality and hint usefulness is still unreliable; adding further sub-steps may boost overall performance but it compounds the evaluation challenge.

Limited Data and Language Coverage. Translating math-word problems is non-trivial: real-world contexts must be preserved, and many cultural references lack direct equivalents (Shi et al., 2022). We therefore rely on the manually curated MGSM dataset, which contains only 250 problems per language across 11 languages. While sufficient for our experiments, this scale limits analyses such as comparing gains across typologically related languages or training models on parallel corpora to induce language-agnostic pedagogy. Expanding high-quality, multilingual datasets, especially for low-resource languages, remains a pressing need.

Acknowledgments

We thank the supercomputing center at MBZUAI for providing resources for this research.

References

Dibyanayan Bandyopadhyay, Soham Bhattacharjee, and Asif Ekbal. 2025. Thinking machines: A survey of llm based reasoning strategies. *arXiv preprint arXiv:2503.10814*.

Luca Benedetto, Giovanni Aradelli, Antonia Donvito, Alberto Lucchetti, Andrea Cappelli, and Paula Buttery. 2024. Using LLMs to simulate students' responses to exam questions. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11351–11368.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Nuo Chen, Zinan Zheng, Ning Wu, Ming Gong, Dongmei Zhang, and Jia Li. 2024. Breaking language barriers in multilingual mathematical reasoning: Insights and observations. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7001–7016, Miami, Florida, USA. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mo Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168.

Nico Daheim, Jakub Macina, Manu Kapur, Iryna Gurevych, and Mrinmaya Sachan. 2024. Stepwise verification and remediation of student reasoning errors with large language model tutors. *Preprint*, arXiv:2407.09136.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

OpenAI Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mkadry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alexander Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alexandre Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, An drey Mishchenko, Angela Baek, Angela Jiang, An toine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, B. Ghorbani, Ben Leimberger, Ben Rossen, Benjamin Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll L. Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Chris Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mély, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Phong Duc Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Hai-Biao Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Pondé de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian D. Kivlichan, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, İbrahim Cihangir Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub W. Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Ryan Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quiñonero Candela, Joe Beutler, Joe Landers, Joel Parish, Jo hannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Joshua Gross, Josh Kaplan, Josh Snyder, Josh Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lil ian Weng, Lindsay McCallum, Lindsey Held, Ouyang Long, Louis Feuvrier, Lu Zhang, Lukasz Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Made laine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Ma teusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Ali Yatbaz, Mengxue Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Castro,

Mikhail Pavlov, Miles Brundage, Miles Wang, Mina Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Na talie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nikolas A. Tezak, Niko Felix, Nithanth Kudige, Nitish Shirish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Phil Tillet, Prafulla Dhariwal, Qim ing Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Raphael Gontijo Lopes, Raul Puri, Reah Miyara, Reimar H. Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Ramilevich Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal A. Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. 2024. Gpt-4o system card. *ArXiv*, abs/2410.21276.

Feihu Jin, Yifan Liu, and Ying Tan. 2024. Zero-shot chain-of-thought reasoning guided by evolutionary algorithms in large language models. *ArXiv*, abs/2402.05376.

Hyunwoo Ko, Guijin Son, and Dasol Choi. 2025. Understand, solve and translate: Bridging the multilingual mathematical reasoning gap. arXiv preprint arXiv:2501.02448.

Ekaterina Kochmar, Dung Do Vu, Robert Belfer, Varun Gupta, Iulian Vlad Serban, and Joelle Pineau. 2022. Automated data-driven generation of personalized pedagogical interventions in intelligent tutoring systems. *International Journal of Artificial Intelligence in Education*, 32(2):323–349.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Jakub Macina, Nico Daheim, Sankalan Pal Chowdhury, Tanmay Sinha, Manu Kapur, Iryna Gurevych, and Mrinmaya Sachan. 2023. Mathdial: A dialogue tutoring dataset with rich pedagogical properties grounded in math reasoning problems. *arXiv preprint arXiv:2305.14536*.

Noburo Matsuda, Michael Yudelson, and Kenneth R. Koedinger. 2013. Simstudent: An agent that learns by tutoring. In *Proceedings of the 16th International Conference on Artificial Intelligence in Education*.

Hunter McNichols, Jaewook Lee, Stephen Fancsali, Steve Ritter, and Andrew Lan. 2024. Can large language models replicate its feedback on open-ended math questions? *arXiv preprint arXiv:2405.06414*.

Jennifer Meyer, Thorben Jansen, Ronja Schiller, Lucas W Liebenow, Marlene Steinbach, Andrea Horbach, and Johanna Fleckenstein. 2024. Using llms to bring evidence-based feedback into the classroom: Ai-generated feedback increases secondary students' text revision, motivation, and positive emotions. *Computers and Education: Artificial Intelligence*, 6:100199.

Benjamin Paaßen, Barbara Hammer, Thomas W. Price, Tiffany Barnes, Sebastian Gross, and Niels Pinkwart. 2017. The continuous hint factory: Providing hints in vast and sparsely populated edit-distance spaces. *Journal of Educational Data Mining*, 9(1):1–35.

Zhuoshi Pan, Yu Li, Honglin Lin, Qizhi Pei, Zinan Tang, Wei Wu, Chenlin Ming, H. Vicky Zhao, Conghui He, and Lijun Wu. 2025. Lemma: Learning from errors for mathematical advancement in llms. *arXiv preprint arXiv:2503.17439*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Zachary A Pardos and Shreya Bhandari. 2024. Chatgpt-generated help produces learning gains equivalent to human tutor-authored help on mathematics skills. *Plos one*, 19(5):e0304013.

Huy Phung, Michael Khosravani, Tuan Nguyen, Zameer Ul Hassan, and Zhihao Wu. 2024. Leveraging gpt-4 tutor model for hint generation and gpt-3.5 student model for hint validation. In *Proceedings of the*

2024 Learning Analytics & Knowledge Conference, Kyoto, Japan.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2022. Language models are multilingual chain-of-thought reasoners. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4780–4797.

Kv Aditya Srivatsa and Ekaterina Kochmar. 2024. What makes math word problems challenging for LLMs? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1138–1148, Mexico City, Mexico. Association for Computational Linguistics.

John Stamper, Tiffany Barnes, Marvin Croy, and Lisa F. Lehman. 2008. The hint factory: Automatic generation of contextualized help for existing computer-aided instruction. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, pages 373–382.

Junior Cedric Tonga, Benjamin Clement, and Pierre-Yves Oudeyer. 2025. Automatic generation of question hints for mathematics problems using large language models in educational technology. In *Proceedings of Large Foundation Models for Educational Assessment*, volume 264 of *Proceedings of Machine Learning Research*, pages 61–102. PMLR.

Leo Törnqvist, Pentti Vartia, and Yrjö O Vartia. 1985. How should relative changes be measured? *The American Statistician*, 39(1):43–46.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D Nguyen. 2025. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*.

Rose Wang, Qingyang Zhang, Carly Robinson, Susanna Loeb, and Dorottya Demszky. 2024a. Bridging the novice-expert gap via models of decision-making: A case study on remediating math mistakes. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2174–2199, Mexico City, Mexico. Association for Computational Linguistics.

Rose E Wang, Ana T Ribeiro, Carly D Robinson, Susanna Loeb, and Dora Demszky. 2024b. Tutor copilot: A human-ai approach for scaling real-time expertise. *arXiv preprint arXiv:2410.03017*.

Murong Yue, Wenhan Lyu, Wijdane Mifdal, Jennifer Suh, Yixuan Zhang, and Ziyu Yao. 2024. Mathvc: An llm-simulated multi-character virtual classroom for mathematics education. In *Proceedings of the 11th ACM Conference on Learning at Scale*.

Feature	Mean Feature Value		T-Statistic	p-value
	GSM8K	MGSM		
Gx_mean_numerical_word_rank	28638.85	28661.13	-0.413	0.680
Gx_word_arg_count	0.64	0.68	-0.407	0.684
Gx_world_knowledge	1.07	1.06	0.140	0.889
Qx_constituency_tree_depth	10.97	10.96	0.044	0.965
Qx_flesch_kinkaid_grade	4.19	4.18	0.071	0.943
Qx_flesch_reading_ease	88.93	89.10	-0.211	0.833
Qx_mean_numerical_word_rank	22739.56	22592.54	0.664	0.507
Qx_mean_word_rank	10664.46	10596.28	0.466	0.641
Qx_multi_np_count	0.44	0.37	0.884	0.377
Qx_np_count	18.52	18.62	-0.208	0.835
Qx_prp_count	1.82	1.90	-0.688	0.492
Qx_sentence_length	3.46	3.44	0.134	0.893
Qx_token_length	67.43	67.49	-0.034	0.973
Qx_unique_np_count	3.50	3.41	0.862	0.389
Qx_word_arg_count	1.11	1.17	-0.599	0.549
Qx_word_length	46.91	46.90	0.008	0.994

Table 7: Pairwise T-Test results between feature level mean values for GSM8K and MGSM.

A Detailed Results

Figure 3 provides a comprehensive summary of the results obtained from our extensive experimental setup.

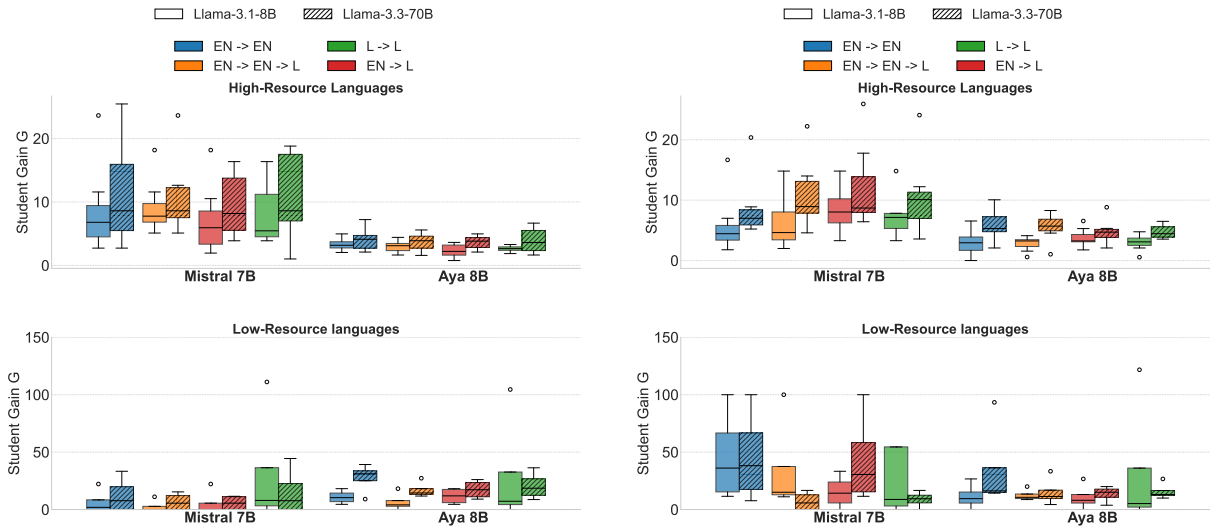
B Dataset Representation

The MGSM (Shi et al., 2022) dataset is built on top of the first 250 math problems from the GSM8K dataset (Cobbe et al., 2021). In order to make sure that this subset is representative of the larger set from GSM8K, we perform a feature level comparison between the two sets. For this, we borrow the feature set from Srivatsa and Kochmar (2024) spanning the phrasing of the math problem in English, the count and nature of math operations and arguments involved in the gold solution, and the count of variables which require world knowledge. After generating the feature values for all English questions from MGSM and GSM8K, we compare their mean values – see Table 7. The low t-statistics and high p-values for corresponding pairwise t-tests indicate that there is not a significant difference between the two sets along any of the features.

C Comparative Analysis of Zero-Shot Performance Across Models

To select our student and tutor models, we initially evaluated five models – `Mistral-7B`, `Aya-8b`, `LLaMA-3.1-8B`,`LLaMA-3.3-70B`, and `Deepseek-R1-LLaMA-distill-70B`[4] – using zero-shot prompting across both setups. We opted for standard zero-shot prompting over zero-shot Chain-of-Thought (CoT) prompting (Jin et al.,

[4]Via Together.ai: https://www.together.ai/models/deepseek-r1-distilled-llama-70b-free



(a) Multilingual prompting

(b) English-only prompting

Figure 3: Relative improvement (Student gain G) in multilingual student-tutor interaction.

2024), as CoT led to correct answers, whereas we sought student models that produce a balanced mix of correct and incorrect responses. Based on these criteria, we selected `Aya-8b` and `Mistral-7B` as student models, as shown in Table 8, which reports their balanced zero-shot accuracy—the baseline for subsequent comparisons. Table 8 also shows that `LLaMA-3.3-70B` achieves the highest accuracy across both setups and outperforms `Deepseek-R1-LLaMA-distill-70B`, particularly with better output structure in low-resource languages. We therefore selected `LLaMA-3.3-70B` and `LLaMA-3.1-8B` as tutor models.

D Implementation Details

We set the temperature to 0 for the student models to ensure deterministic outputs, while the tutor models use a temperature of 1 to encourage diverse hint generation. GPT-4o (Hurst et al., 2024) was used with a temperature of 0 to evaluate the correctness of both the initial and revised answers by comparing them to the gold answer, as prior work has shown that temperatures above 0.2 can lead to unreliable results (Tonga et al., 2025). For the tutor models, we employed `LLaMA-3.3-70B` and `LLaMA-3.1-8B`. For the student models, we selected a monolingual model, `Mistral-7B`, and a multilingual model, `Aya-8b`. All models and their corresponding reproduction links are presented in Table 9.

E Prompts

In this section, we present the prompts used in our experiments.

E.1 Student prompts

Figure 4 shows the base prompt for generating a candidate solution, while Figure 5 displays the prompt for revising an initial answer via the student model. These prompts are used as-is in the English-only prompting setup. For the multilingual prompting setup, they are translated into the 10 target languages of the MGSM dataset.

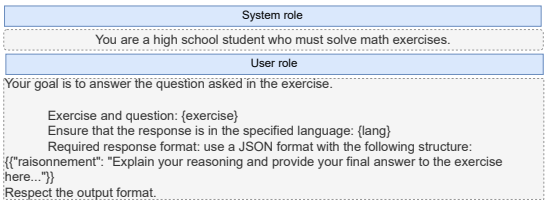


Figure 4: Prompt for generating a candidate solution.

E.2 Hint generation prompt

Figure 6 shows the base prompt used to generate a hint via the tutor model. For strategies where the tutor is prompted in English, the prompt is used as-is, with the variable ***hint_lang*** in the prompt replaced by either "English" (EN) or the exercise language (L), depending on the desired hint language. For strategy, where the tutor is prompted in the language of the exercise L, the prompt is translated into that language, and ***hint_lang*** is again set

English only prompting: when student input prompt is in English language					
Languages	**Llama 3.1-8B**	**Aya-8b**	**Mistral-7B**	**Llama-3.3-70B**	**Deepseek-R1-Llama-distill-70B**
English	85.2	77.2	60.8	93.2	85.2
Spanish	76.8	72.0	46.0	86.0	80.8
French	74.8	68.0	44.8	86.4	78.4
German	74.8	68.0	40.0	86.8	82.4
Russian	72.4	76.8	43.6	90.8	80.8
Chinese	72.4	67.2	36.0	88.0	84.0
Japanese	54.8	61.6	21.6	84.8	83.2
Thai	64.8	21.2	20.8	87.6	88.4
Swahili	60.8	9.2	3.6	84.8	79.2
Bengali	60.0	28.0	9.6	85.6	79.2
Telugu	53.2	6.0	0.4	81.6	58.0
AVG	68.2%	50.5%	29.7%	**86.9**%	80.0%
Multilingual prompting: when student input prompt is in native language					
Languages	**Llama 3.1-8B**	**Aya-8b**	**Mistral-7B**	**Llama-3.3-70B**	**Deepseek-R1-Llama-distill-70B**
English	85.2	77.6	60.4	93.6	82.8
Spanish	76.8	78.4	41.2	89.6	82.0
French	74.8	73.2	44.0	80.0	77.6
German	74.8	73.6	40.4	88.8	80.0
Russian	72.4	76.0	39.2	93.2	82.8
Chinese	72.4	72.0	38.0	88.0	86.8
Japanese	54.8	64.4	22.0	84.8	83.2
Thai	64.8	18.4	10.4	86.8	83.2
Swahili	60.8	8.8	3.6	87.6	82.0
Bengali	60.0	20.4	9.2	85.6	81.6
Telugu	53.2	4.4	0.4	84.8	75.2
AVG	68.2	51.5	28.1	**87.5**	81.6

Table 8: Zero-shot accuracy (%) of language models across languages. These zero-shot scores serve as the baseline for each model. The table shows results for both English-only prompting (top) and multilingual prompting (bottom) setups. `Llama-3.3-70B` consistently achieves the highest accuracy across most languages, demonstrating superior cross-lingual capabilities in zero-shot settings.

Model	**Reproduction Link**
`Llama-3.3-70B`	`https://www.together.ai/models/llama-3-3-70b-free`
`Mistral-7B`	`https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3`
`Llama-3.1-8B`	`https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct`
`Aya-8b`	`https://huggingface.co/CohereLabs/aya-expanse-8b`
GPT-4o	gpt-4o-2024-08-06

Table 9: Models used in our experiments with their corresponding links for reproducibility.

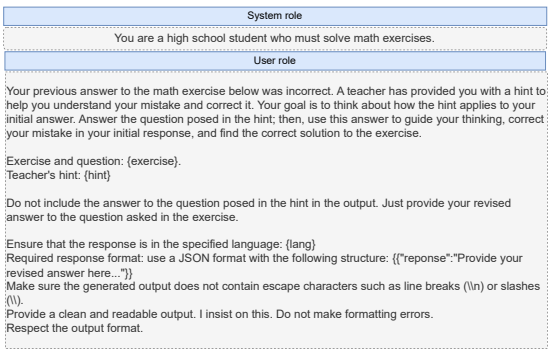


Figure 5: Prompt for revising the initial candidate solution

based on whether the hint should be in English or the exercise language.

E.3 Prompts for Evaluating Student Outputs

Figure 7 shows the prompt used to evaluate the initial candidate solution against the gold solution, while Figure 8 presents the prompt for evaluating the revised solution via GPT-4o, after a hint is provided. We adopt the evaluation prompts from Tonga et al. (2025), but omit their error type categorization for the initial candidate solution, as it is beyond the scope of our work. However, we retain their approach of classifying hints as good (if the

System role

You are an expert in teaching mathematics.

User role

```
Your goal is to provide {num_indices} clear and relevant hints to help the student correct their errors and improve
their answers in math exercises.
This hint should be in the form of a question. Moreover, this hint should not include the correct answer to the
exercise or fragments of the correct answer.

Exercise and question: {exercise}.
The correct answer to the exercise: {answer}
The student's reasoning, including their final answer to the exercise: {gpt_reasoning}
Language of the {num_indices} hints: {hint_lang}

Consider the following aspects to generate the hint:
    - Reasoning
    - Method
    - Concept application
    - Calculations
    - Problem interpretation

JSON output format: {{"indice": ["indice1", "indice2", ..., "indice{num_indices}"]}}. Do not number the hints.

Please generate exactly {num_indices} hint(s), no more. Follow this condition strictly.
The {num_indices} hints should be in the specified language: {hint_lang}
Ensure that the generated output does not contain escape characters such as new lines (\\n) or slashes (\\).
Provide a clean and readable output. I insist on this. Do not make formatting errors.
Follow the output format strictly. I emphasize that the hint should not be numbered and must be in the form of a
question.
```

Figure 6: Prompt used by the tutor model to generate hints.

revision is correct) or bad (if incorrect) to track hint effectiveness.

System role

You are an expert in math teaching.

User role

```
Your task is to verify whether a student's answer to a math exercise is correct or not by comparing it with the provided correct answer.
Exercise and question: {exercise}
The correct answer to the exercise: {answer}
The student's reasoning containing their final answer to the exercise: {reasoning}
Extract the student's final answer from their reasoning, based on the question asked in the exercise, in order to compare it with the
correct answer provided for the exercise.Categorize the student's error. Here are some error categories and examples. You can add other
error categories. If the reasoning contains multiple errors, it is important to list all the errors present.
Specify each error distinctly, even if they belong to different categories or are combined.
    1) Comprehension error: The student does not clearly understand the problem or the given instructions.
    Example: Misreading a problem and confusing the given data.
    2) Partial answer: The student provides part of the expected answer but fails to complete it correctly.
    Example: In an equation with two variables, the student finds the value of one variable but forgets to find the value of the other.
    3) Term grouping error: The student incorrectly combines or groups terms in a mathematical expression.
    Example: When simplifying the expression 3x + 2x + 5, the student combines the terms 3x and 2x to obtain 5x^2 instead of 5x.
    4) Simplification error: The student incorrectly simplifies a mathematical expression.
    Example: When simplifying 6x/2, the student divides the numerator and denominator by x instead of 2, resulting in an incorrect
simplification of 6/2x.
    5) Calculation error: The student performs mathematical operations incorrectly.
    Example: When multiplying 7 by 8, the student gets 54 instead of 56.
    6) Incorrect substitution error: The student substitutes an incorrect value in an expression or equation.
    Example: In the equation 2x + 3y = 10, the student substitutes x = 4 instead of y = 2, leading to an incorrect solution.
    7) Interpretation error: The student misinterprets the instructions or data of a problem.
    Example: In a probability problem, the student confuses the probability of event A with that of the complementary event of A.
    8) Algebraic error: The student makes a mistake in algebraic manipulations, such as distribution, factoring, or equation solving.
    Example: When solving 2(x + 3) = 10, the student incorrectly divides 10 by x + 3 instead of 2, leading to an incorrect answer.
1) If the student's final answer does not match the correct answer, categorize the type of error by placing it in "type_d'erreur" field and do
not put anything in the "bonne_reponse" field.
2) If the student's final answer matches the correct answer, place the student's answer in the "bonne_reponse" field.
Provide the output in a JSON format with the following structure: {{"type_d'erreur": "", "bonne_reponse": ""}}
Respect the output format, the evaluation criteria, and your role. Do not add anything else.
```

Figure 7: Prompt employed by GPT-4o to assess candidate answer correctness.

System role

You are an expert in math teaching.

User role

```
Your task is to verify whether a student's revised answer to a math exercise is correct or not by
comparing it with the provided correct answer.
The correct answer to the exercise: {answer}
The student's revised answer: {revised_response}
The hint: {hint}

    1) If the student's revised answer does not match the correct answer, place the hint in the
"mauvais_hint" field in the output.

    2) If the student's revised answer matches the correct answer, place the hint in the "bon_hint"
field in the output.

    3) If the given hint contains the correct answer to the exercise, place the hint in the
"mauvais_hint" field in the output.

Provide the output in a JSON format with the following structure: {{"bon_hint": "", "mauvais_hint": ""}}
Make sure the generated output does not contain escape characters such as line breaks (\\n) or
slashes (\\).
Provide a clean and readable output. I insist on this. Do not make formatting errors.
Respect the output format, the evaluation criteria, and your role. Do not add anything else.
```

Figure 8: Prompt employed by GPT-4o to assess the correctness of the revised solution and categorize hints.

Axis	Values	Count
Languages	en, bn, de, es, fr, ja, ru, sw, te, th, zh	11
Student Prompts	English-only, Multilingual	2
Student Models	`Mistral-7B`, `Aya-8B`	2
Hint Strategies	EN→EN, EN→EN→L, L→L, EN→L	4
tutor Models	`LLaMA-3.1-8B`, `LLaMA-3.3-70B`	2
Total Configs	$11 \times 2 \times 2 \times 4 \times 2$	**352**

Table 10: Overview of the experimental space.

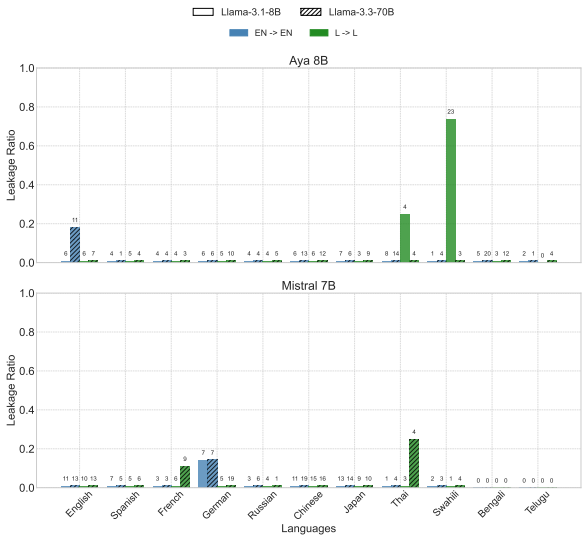


Figure 9: Leakage ratio per language; numbers above bars indicate total helpful hints.

F Analysis: Student gains across languages for the two different student models with `LlaMA-3.3-70B`

Table 13 and Table 12 show the student gains of `Aya-8b` and `Mistral-7B`, respectively, after giving a hint, when we used `LlaMA-3.3-70B` as the tutor model.

Language	BLEU	Language	BLEU
Spanish	62.7	Chinese	40.1
French	54.7	Japanese	45.6
German	46.7	Thai	40.0
Russian	49.3	Swahili	45.5
Bengali	43.0	Telugu	39.5

Table 11: BLEU scores by language

		English-only prompting					Multilingual prompting			
		EN→EN	EN→EN→L	EN→L	L→L		EN→EN	EN→EN→L	EN→L	L→L
HRLs	English	7.9	7.9	7.9	5.9	English	8.6	8.6	6.0	8.6
	Spanish	5.2	7.8	8.7	10.4	Spanish	4.9	7.8	3.9	5.8
	French	5.4	8.9	8.0	3.6	French	2.7	7.3	8.2	8.2
	German	7.0	14.0	10.0	8.0	German	11.9	11.9	13.9	18.8
	Russian	6.4	4.6	6.4	10.1	Russian	6.1	5.1	5.1	1.0
	Chinese	8.9	12.2	17.8	12.2	Chinese	20.0	12.6	13.7	16.8
	Japanese	20.4	22.2	25.9	24.1	Japanese	25.5	23.6	16.4	18.2
LRLs	Thai	7.7	11.5	11.5	7.7	Thai	15.4	15.4	11.5	15.4
	Swahili	55.6	0.0	44.4	11.1	Swahili	33.3	11.1	11.1	44.4
	Bengali	20.8	16.7	16.7	16.7	Bengali	0.0	0.0	0.0	0.0
	Telugu	100.0	0.0	100.0	0.0	Telugu	0.0	0.0	0.0	0.0

Table 12: Student gains G (%) of `Mistral-7B` after receiving a hint from `LlaMA-3.3-70B` across English-only and multilingual prompting setups.

		English-only prompting					Multilingual prompting			
		EN→EN	EN→EN→L	EN→L	L→L		EN→EN	EN→EN→L	EN→L	L→L
HRLs	English	5.7	5.7	3.1	4.1	English	4.1	4.1	3.1	3.6
	Spanish	5.0	7.2	5.0	4.4	Spanish	5.1	5.1	2.6	2.0
	French	8.8	6.5	8.8	5.9	French	2.2	2.7	3.8	1.6
	German	5.3	5.3	4.7	5.3	German	3.3	2.7	4.9	5.4
	Russian	2.1	1.0	2.1	3.6	Russian	2.1	1.6	2.1	2.6
	Chinese	10.1	8.3	5.3	3.6	Chinese	7.2	3.9	3.9	6.7
	Japanese	4.5	4.5	4.5	6.5	Japanese	4.3	5.6	5.0	5.6
LRLs	Thai	15.1	11.3	3.8	13.2	Thai	30.4	15.2	26.1	8.7
	Swahili	17.4	4.3	13.0	13.0	Swahili	31.8	13.6	22.7	13.6
	Bengali	14.3	11.4	17.1	10.0	Bengali	39.2	11.8	11.8	23.5
	Telugu	93.3	33.3	20.0	26.7	Telugu	9.1	27.3	9.1	36.4

(a) English-only prompting

(b) Multilingual prompting

Table 13: Student gain G (%) of `Aya-8b` after receiving a hint from `LlaMA-3.3-70B` in the two setups.

Beyond Accuracy: Diagnosing Algebraic Reasoning Failures in LLMs Across Nine Complexity Dimensions

Parth Shrikant Patil[1], **Yash Sinha**[2], **Murari Mandal**[3], **Dhruv Kumar**[2]
[1]Nanyang Technological University (NBS)
[2]Birla Institute of Technology and Science, Pilani
[3]Kalinga Institute of Industrial Technology (KIIT), Bhubaneswar
parth010@e.ntu.edu.sg

Abstract

Aggregate mathematical accuracy does not usually identify the source of a model failure. In algebraic reasoning, errors may arise from nesting depth, operator identity, simultaneous intermediate values, branch-local composition, repeated-token counting, sequential dependence, or numeric magnitude. We introduce a nine-dimension algebraic complexity framework that converts these factors into controlled diagnostic variables. For each dimension, a parametric generator increases one target variable while holding the remaining variables at minimum values. Every retained problem is verified by a computer algebra system. We evaluate eight models, including six standard instruction-tuned models and two reasoning-specialised models. The results show that standard models exhibit a scale-limited D4 working-memory collapse in the 20–40 parallel-branch region. Notably, Qwen3 235B does not escape the transition band observed for Llama 3 8B, despite its much larger parameter scale. Reasoning-specialised models shift this limit to higher levels, but they do not remove it. D5 branch-local composition and D7 repeated-token counting produce distinct failures that are not reducible to aggregate length or depth.

1 Introduction

Mathematical reasoning is a central evaluation setting for large language models (LLMs). Existing benchmarks include grade-school word problems (Cobbe et al., 2021), competition mathematics (Hendrycks et al., 2021), graduate-level applied problems (Fan et al., 2024), and symbolic computation (Lample and Charton, 2020; Saxton et al., 2019). These benchmarks are valuable for comparing models, but their aggregate scores often conceal the cause of an incorrect response. A model may fail because an expression contains deeply nested subexpressions, an unfamiliar operator, many simultaneous intermediate values, repeated terms that must be counted, or a long chain in which early errors compound.

This attribution problem becomes more important as static benchmarks saturate or become contamination-sensitive. Dynamic variants such as GSM-Symbolic (Mirzadeh et al., 2025) and Putnam-AXIOM (Gulati et al., 2025) show that surface changes can alter performance. Surveys of mathematical reasoning failures also emphasise that LLM errors are heterogeneous rather than a single capability gap (Song et al., 2026). Prior studies isolate individual mechanisms. Multiplication can induce $O(n^2)$ cross-token interactions (Dziri et al., 2023). Nesting depth causes recursive propagation losses (Saxton et al., 2019). Counting is sensitive to tokenisation and positional structure (Chang and Bisk, 2024; Zhang et al., 2024; Yehudai et al., 2024). Compositional reasoning can also break when subproblems are chained (Zhao et al., 2024; Hosseini et al., 2024). However, existing work does not yet provide a unified algebraic benchmark in which these mechanisms are separated, scaled systematically, and compared across model families.

We address this gap by defining nine algebraic complexity dimensions. They are syntactic length (D1), tree depth (D2), operator hardness (D3), working-memory load (D4), compositional branching (D5), solution ambiguity (D6), counting load (D7), sequential chain length (D8), and numeric magnitude (D9). Each dimension is implemented as a generator control parameter. Increasing D4, for example, adds independent single-digit products that must be stored before summation. Increasing D5 leaves the number of branches fixed but makes each branch internally more compositional. Increasing D7 repeats an identical operand many times so that the active burden is enumeration rather than arithmetic. The appendix provides full definitions, examples at multiple levels, and the structural coupling analysis (Appendix A (dimension guide); Appendix A (example items); Appendix D (cou-

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 216–239
October 29, 2026 ©2026 Association for Computational Linguistics

plings)).

Our main empirical result is that standard instruction-tuned models share a scale-limited D4 working-memory failure. All standard models remain strong at small branch counts, but every standard model in the updated D4 battery reaches at least one zero-accuracy point by 40 branches. A 235B standard model does not escape the same transition band as the open 8B/12B baselines, which suggests that parameter count alone does not create register-like algebraic state. Reasoning-specialised models substantially shift D4 and D5 thresholds, but the extended batteries still locate failure. Qwen QwQ fails at 500 D4 branches and 50 D5 operations per branch, while DeepSeek R1 0528 fails at 3000 D4 branches and 70 D5 operations per branch. Counting remains especially idiosyncratic: Claude 3.5 Haiku stays perfect through $K = 300$, but DeepSeek R1 0528 fails at $K = 300$ and several models show non-monotonic recoveries.

Contributions. We make three contributions.

- **A nine-dimension algebraic diagnostic framework.** We operationalise nine structurally distinct algebraic complexity dimensions and connect each one to a documented LLM failure mechanism.
- **Automated, systematically scalable problem generation.** We introduce a parametric generator-and-verifier pipeline that raises the difficulty of one dimension at a time. The pipeline verifies every retained problem with a computer algebra system, avoiding human annotation and fixed-corpus ceilings.
- **Cross-model diagnostic evaluation.** We evaluate six standard instruction-tuned models and two reasoning-specialised models under the same controlled batteries, identifying which failures are shared, which are shifted by reasoning traces, and which remain model-specific.

2 Related Work

Mathematical benchmarks and attribution. GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), HardMath (Fan et al., 2024), and symbolic-mathematics benchmarks (Lample and Charton, 2020; Saxton et al., 2019) evaluate broad mathematical competence. These benchmarks usually combine several structural pressures inside each item, which makes failure attribution difficult. GSM-Symbolic (Mirzadeh et al., 2025) and Putnam-AXIOM (Gulati et al., 2025) show that benchmark performance can change under controlled variants. However, they do not decompose algebra into independently scaled failure dimensions. Our benchmark uses controlled generation to produce per-dimension failure curves rather than a single aggregate score.

Automated problem generation. Prior work has generated algebra variants through syntactic transformation and verification (Singh et al., 2012) and procedural elementary-math templates (Xu et al., 2021). Other systems use large-scale LLM-assisted data generation (Tang et al., 2024; Manem et al., 2025; Chen et al., 2025). These systems primarily support practice or training. Our pipeline is diagnostic. Each generator is tied to a named complexity variable, and each level is used to estimate a failure threshold.

Controlled complexity and reasoning models. Length-generalisation studies show that models may solve short algorithmic instances while failing on longer ones (Anil et al., 2022). CLRS-Text varies problem size and trace length in algorithmic reasoning (Markeeva et al., 2024); seqBench varies sequential depth and finds collapse as action length increases (Ramezanali et al., 2025). Recent trace-aware benchmarks further isolate depth-scaled sequential decay across task regimes and intermediate-state execution errors in algorithmic reasoning (Chapra et al., 2026; Kapoor et al., 2026). Reasoning-specialised models such as DeepSeek R1 (DeepSeek-AI, 2025) and Qwen QwQ (Alibaba Cloud, 2025) improve aggregate mathematical scores through extended reasoning traces. However, aggregate gains do not reveal which structural failures are eliminated and which are merely displaced. Controlled puzzle studies similarly find thresholded collapse even for reasoning models (Shojaee et al., 2025). We bring this controlled-complexity view to CAS-verified algebra.

3 Benchmark Design

The benchmark follows two principles. First, each dimension must describe a distinct algebraic pressure rather than a surface statistic alone. Second, only one variable should change inside a battery. The generator fixes all non-target dimensions at their minimum viable values, then increases the tar-

get variable across a level schedule chosen to reveal both gradual degradation and sharp thresholds.

Table 1 summarises the dimensions. The table is organised around failure mechanisms rather than surface difficulty labels. The same expression can be long because it has many repeated terms, a deep tree, or a long dependent chain. These cases impose different computational burdens and should not be collapsed into a single category of difficult algebra. The benchmark therefore treats each dimension as a testable hypothesis about why a model might fail.

Length, depth, and sequential dependence. D1 measures syntactic length in prefix notation. It is a useful stressor, but it is also the most confounded dimension because many other structures make expressions longer. D2 measures tree depth, where a model must resolve deeply nested partial results and propagate them back to the root. D8 measures sequential chain length, where only one live state is carried forward but each step depends on the previous result. D2 and D8 can both create long expression spines. Nevertheless, they stress different computations: recursive return through a tree versus repeated state update.

Operators and numeric magnitude. D3 isolates operator identity by holding depth, branch count, counting load, and operand size fixed while moving across an ordered operator ladder. This setup tests whether an error comes from the symbolic operation itself rather than from surrounding structure. D9 fixes the operator to multiplication and increases digit count, targeting partial-product accumulation and representation effects. D9 should therefore not be interpreted simply as a test of large numbers. It tests large operands under an operator whose cross-digit interactions grow rapidly.

Parallel memory and branch-local composition. D4 and D5 are the two most important structural dimensions to separate. D4 increases the number of independent products that must coexist before final summation. The arithmetic inside each branch is simple, so the intended burden is horizontal storage of many intermediate values. D5 keeps the number of branches fixed at three and instead increases the number of operations inside each branch. The intended burden is local composition before each branch resolves. A model can pass one of these dimensions and fail the other, which is why both are necessary.

Strategy choice and counting. D6 changes the algebraic problem type, from canonical equations to forms with several plausible solution routes. It tests strategy selection and form familiarity more than raw computation. D7 repeats an identical operand K times and asks the model to enumerate the repetitions exactly. This dimension is intentionally simple arithmetically: the point is to test cardinality tracking over repeated tokens. D7 is especially important because its failures are often not predicted by depth, branch count, or model scale.

This decomposition makes the benchmark diagnostic rather than merely difficult. A D4 failure indicates weakness in simultaneous intermediate-state management. A D7 failure indicates weakness in repeated-token cardinality. A D2 or D8 failure indicates difficulty with recursive or sequential state tracking. The appendix provides generated examples at multiple levels for every dimension (Appendix A (example items)).

4 Isolation, Scaling, and Verification

Each battery is generated under a one-active-variable protocol with two stages. First, we define the minimum viable template for the target dimension. Operands are single-digit when possible, operators are low-hardness unless D3 is active, branch count is minimal unless D4 is active, and sequential dependence is avoided unless D8 is active. Second, after this controlled baseline is fixed, we raise only the target variable. This design turns the benchmark into a set of controlled interventions on algebraic structure, rather than a collection of increasingly complicated problems.

Isolation examples. In D4, the generator creates a sum of independent single-digit products. Raising the level from small branch counts into the extended hundreds- and thousands-branch regime does not introduce harder operators or larger operands. It only increases the number of branch outputs that must be retained before the final sum. This distinction matters because the results show a standard-model transition band near moderate branch counts, while the reasoning-model extensions require much larger D4 levels before the next collapse appears. In D5, the generator fixes exactly three branches joined by two root operators and increases the operations inside each branch. In D7, the operand and arithmetic operation are fixed while the repeated-term count K grows. In D8, each level adds a dependent update step while keeping the live state count at one. These

Dim.	Name	What Is Varied	Failure Mechanism Isolated
D1	Syntactic length	Prefix-token count in simple expression chains.	Length extrapolation when arithmetic stays easy.
D2	Tree depth	Longest root-to-leaf path in a nested expression.	Recursive partial-result propagation.
D3	Operator hardness	Operator identity from basic arithmetic to powers and trigonometric forms.	Operator-specific symbolic competence.
D4	Working memory	Number of independent products summed after parallel evaluation.	Simultaneous storage of many intermediate values.
D5	Branch complexity	Operations inside each of three fixed branches.	Branch-local composition before branch outputs combine.
D6	Solution ambiguity	Algebraic problem type and number of plausible solution routes.	Strategy selection rather than raw arithmetic.
D7	Counting load	Number of identical repeated terms K.	Cardinality tracking over repeated tokens.
D8	Sequential chain length	Number of strictly dependent update steps.	Cascading error along a one-state chain.
D9	Numeric magnitude	Digit count in multiplication operands.	Cross-digit partial-product accumulation.

Table 1: The nine algebraic complexity dimensions used in the benchmark. Full definitions, examples, level schedules, and literature grounding are provided in Appendix A (dimension guide) and Appendix A (example items).

choices make a drop in accuracy interpretable.

Systematic scaling. The same generator can raise a dimension ceiling without redesigning the benchmark. D4 is scaled through 2, 4, 8, 12, 20, 25, 30, 40, 50, 60, 75, 100, 150, 200, 500, 1000, and 3000 branches. D5 is scaled from 0 to 100 operations per branch. D7 is scaled from $K = 5$ to $K = 300$ repeated terms. D8 is scaled from 1 to 500 dependent steps. Smaller increments are placed around expected transition regions, while wider jumps probe extrapolation after standard models have already collapsed. The complete schedules are listed in Appendix A (level schedules).

Problems are represented internally in Polish prefix notation because token count, tree depth, branching, and chain length are directly measurable from the sequence. Before prompting, every item is converted to standard ASCII infix notation. Each generated candidate is submitted to a computer algebra system (CAS). Candidates are rejected if they violate dimension controls, produce ambiguous solver output, exceed the CAS timeout, raise a solver exception, or yield a non-finite answer. This rejection-sampling loop allows the benchmark to be generated automatically while retaining deterministic, finite ground truth. The same automation makes the benchmark extensible: when a model saturates a level, the researcher can raise a generator parameter rather than manually authoring a new test set.

The dimensions are isolated as generator controls, not claimed to be mathematically independent in every surface statistic. Some co-movement is unavoidable: increasing D4 also lengthens the expression, D5 can increase local depth, and D8 produces both a long chain and a deep expression spine. We therefore analyse directional couplings separately in Appendix D (couplings). This lets the main analysis distinguish a target mechanism from downstream surfaces such as length.

5 Experimental Setup

We evaluate eight models: GPT-4o Mini, Claude 3.5 Haiku, Qwen3 235B, DeepSeek V3, Gemma 3 12B, Llama 3 8B, DeepSeek R1 0528, and Qwen QwQ. The first six are standard instruction-tuned models that span hosted proprietary systems, open 8B/12B models, and large mixture-of-experts models. The final two are reasoning-specialised models with extended reasoning traces.

Table 3 summarises the model coverage. The model set is designed for diagnostic comparison rather than leaderboard ranking. First, the open 8B/12B models and Qwen3 235B test whether failure thresholds shift smoothly with published parameter scale, while hosted proprietary systems provide contemporary API baselines without relying on undisclosed parameter counts. Second, decoder-only and mixture-of-experts models test whether architectural family changes the curve shape. Third, DeepSeek R1 0528 and Qwen QwQ test whether extended reasoning traces eliminate a failure mode or instead push it to larger generated levels. For this reason, the D4 and D5 reasoning extensions are essential. Without raising the ceiling beyond standard-model collapse, reasoning models would appear to solve the dimension.

Each model is evaluated on the same 50 CAS-verified problems for every evaluated dimension level, generated with seed 42. This sample size reflects a practical constraint rather than a theoretical optimum. The evaluation becomes costly at high levels because the prompt size and reasoning burden grow with the active dimension. For example, large D4 settings require a model to process hundreds or thousands of simultaneous products before producing the final sum, which increases input cost, output

Dim.	Scaling Intervention	What Is Held Fixed	Interpretation of Failure
D2	Increase nesting depth from shallow expressions to depth 50 extensions.	Operator family, operand size, and branch width.	Failure indicates nested partial-result propagation limits.
D4	Increase parallel products from $K = 2$ to $K = 3000$.	Each branch remains a trivial single-digit product.	Failure indicates inability to retain many simultaneous intermediate values.
D5	Increase branch-internal operations from 0 to 100.	Exactly three branches and two root operators.	Failure indicates local compositional breakdown before branch outputs combine.
D7	Increase identical repeated terms from $K = 5$ to $K = 300$.	Repeated operand and final arithmetic form.	Failure indicates cardinality/enumeration difficulty rather than arithmetic difficulty.
D8	Increase dependent chain length from 1 to 500 steps.	One live state, small operands, and no parallel branches.	Failure indicates cascading sequential-error accumulation.
D9	Increase multiplication operands from 1 to 15 digits.	Multiplication operator and minimal expression structure.	Failure indicates partial-product and numeric-representation limits.

Table 2: Representative scaling interventions in the one-active-variable protocol. The full nine-dimension schedule is reported in Appendix A (level schedules).

Model	Scale	Type	Architecture	Diagnostic Role
GPT-4o Mini	Undisclosed	Standard	Undisclosed	Hosted small instruction model.
Claude 3.5 Haiku	Undisclosed	Standard	Undisclosed	Hosted small instruction model with strong D7 behaviour.
Qwen3 235B	235B total 22B active	Standard	MoE decoder	Large standard model for scale comparison.
DeepSeek V3	671B total 37B active	Standard	MoE decoder	Large MoE standard model.
Gemma 3 12B	12B	Standard	Decoder-only	Mid-size open-weight model.
Llama 3 8B	8B	Standard	Decoder-only	Small open-weight baseline.
DeepSeek R1 0528	671B total 37B active	Reasoning	MoE + CoT	Reasoning-specialised model with extended traces.
Qwen QwQ	32.5B	Reasoning	Decoder + CoT	Reasoning-specialised model with extended traces.

Table 3: Model coverage in the cross-model evaluation. The selection is designed to compare scale, architecture family, and reasoning-style inference under the same dimension-controlled batteries. Proprietary hosted model sizes are not publicly disclosed; for MoE models, total and active denote total parameters and active parameters per forward pass.

length, and failure-analysis time. We therefore use a fixed 50-item protocol to keep the comparison controlled across models while retaining enough instances to expose threshold behaviour.

Models are queried at temperature zero with no self-consistency, majority voting, tool use, or human correction. The prompt asks for step-by-step reasoning and a final line in the form `ANSWER: value`. Correctness is computed against the CAS answer using the more permissive of $\pm 0.5\%$ relative error or ± 0.05 absolute error. API errors and CAS-generation errors are tracked separately and excluded from accuracy denominators. Prompt templates and example generated items are given in Appendix B.2 (prompt template) and Appendix A (example items).

6 Results

Figure 1 summarises the first zero-accuracy point for D4, D5, and D7. This view makes the main threshold structure visible without compressing full heatmaps into the main text. D4 has a tightly clustered standard-model collapse, D5 shows branch-local degradation even with fixed branch count, and D7 reveals a different counting profile in which Claude 3.5 Haiku does not reach a zero-accuracy point within the evaluated range. Because a first-zero plot necessarily compresses non-monotonic recoveries, Table 5 states those recoveries explicitly and the appendix provides the full heatmaps.

Table 5 complements the threshold plot by reporting the exact first-zero levels and the observed D7 recoveries. D4 thresholds cluster tightly for standard models even though the models differ substantially in scale and architecture. D5 thresholds

Dim.	Diagnostic Role	Standard-Model Pattern	Reasoning-Model Pattern
D2	Vertical nesting.	Accuracy remains high through shallow depths and collapses for most standard models by depths 7–8.	Threshold moves outward; DeepSeek R1 0528 still degrades at extreme depths.
D4	Horizontal working memory.	Every standard model reaches at least one zero-accuracy point by 40 branches, despite isolated recoveries.	Qwen QwQ fails at 500 branches; DeepSeek R1 0528 fails at 3000.
D5	Branch-local composition.	Standard models separate by 3–8 operations per branch and all reach 0% by 50.	Qwen QwQ fails at 50 operations per branch; DeepSeek R1 0528 fails at 70.
D7	Repeated-token counting.	Highly model-specific: Claude stays perfect, Llama collapses early, and several models recover non-monotonically.	DeepSeek R1 0528 stays perfect through $K = 200$ but fails at $K = 300$; Qwen QwQ fails from $K = 60$ with a partial recovery at $K = 100$.
D8	Sequential dependence.	Chains become destructive after the 10-step region for most standard models.	DeepSeek R1 0528 remains perfect; Qwen QwQ eventually fails at 300 and 500 steps.

Table 4: Main diagnostic findings for the dimensions that most clearly separate model behaviour. Full heatmaps, all nine dimensions, and detailed tables are in Appendix C (heatmaps) and Appendix C (full tables).

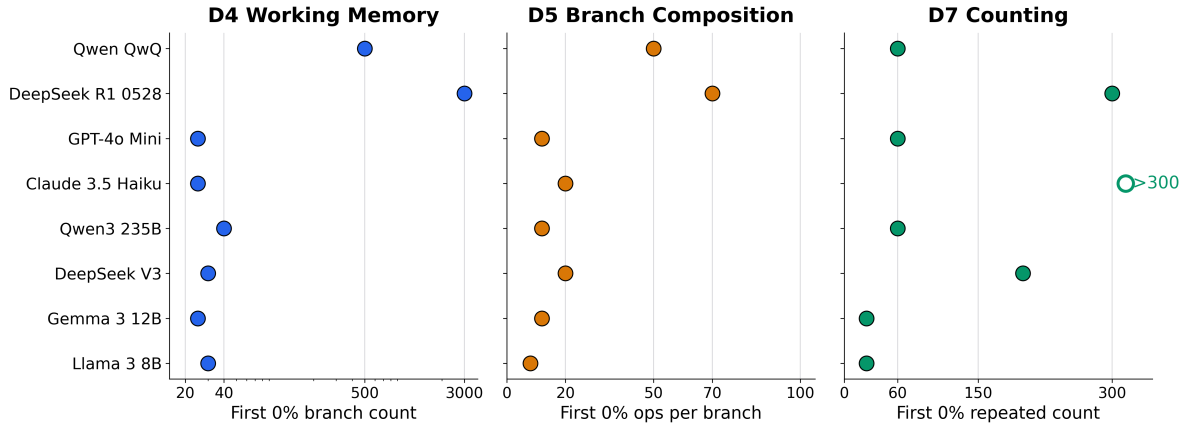


Figure 1: Failure-threshold summary for the three main diagnostic dimensions. Each dot marks the first zero-accuracy level observed for a model: D4 uses the number of parallel branches, D5 uses operations per branch, and D7 uses repeated-term count K. Larger x-axis values indicate a later collapse. The open D7 marker for Claude 3.5 Haiku indicates that no zero-accuracy point appears through $K = 300$. Full heatmaps and detailed tables are provided in Appendix C (heatmaps) and Appendix C (full tables).

are lower for smaller models but remain finite for every standard model. D7 follows a different pattern. The strongest standard model on D7, Claude 3.5 Haiku, is perfect throughout, while several larger or reasoning-specialised systems fail and partially recover. The D7 profile therefore motivates treating counting as a distinct diagnostic axis rather than a side effect of length.

D4 exposes a scale-limited working-memory bottleneck. D4 is the clearest test of simultaneous intermediate-state storage. The threshold plot in Figure 1 shows that the first zero-accuracy D4 point for every standard model lies between 25 and 40 branches. The full D4 heatmap in Figure C.2 gives the surrounding curve: all standard models are perfect through 8 branches, but by 12 branches Gemma 3 12B and Llama 3 8B have slipped to 75%, and the shared failure band appears from 20 to 40 branches. GPT-4o Mini reaches 0% at 25 branches, Qwen3 235B reaches 0% at 40, DeepSeek V3 reaches 0% at 30, and every standard model has at least one zero point by 40. Some models later recover at isolated levels, such as Claude 3.5 Haiku at 40, Qwen3 235B at 50, and DeepSeek V3 at 40, 50, 60, or 150. However, none sustains high-branch reliability. The 29× parameter gap between Qwen3 235B and Llama 3 8B therefore does not translate into a proportionally shifted D4 threshold.

Reasoning-specialised models change the curve but do not remove the cliff. Qwen QwQ remains at 100% through 200 branches except for a 75% point at 40, then falls to 0% at 500, 1000, and 3000 branches. DeepSeek R1 0528 remains perfect through 1000 branches and collapses at 3000. This supports a threshold-shift interpretation: extended reasoning traces act like an external scratchpad, but sufficiently large parallel state still defeats the model.

Model	D4 First 0%	D5 First 0%	D7 Counting Pattern
GPT-4o Mini	25 branches	12 ops/branch	0% at $K = 60$, recovery at $K = 100$, then 0% from $K = 150$.
Claude 3.5 Haiku	25 branches	20 ops/branch	No collapse through $K = 300$; remains 100% at every D7 level.
Qwen3 235B	40 branches	12 ops/branch	0% at $K = 60$, partial recovery at $K = 100$, then 0% from $K = 150$.
DeepSeek V3	30 branches	20 ops/branch	Strong through $K = 150$, then 0% at $K = 200$ and $K = 300$.
Gemma 3 12B	25 branches	12 ops/branch	0% at $K = 25$, recovery at $K = 60$, then 0% from $K = 100$.
Llama 3 8B	30 branches	8 ops/branch	Earliest sustained D7 collapse; 0% from $K = 25$ onward.
DeepSeek R1 0528	3000 branches	70 ops/branch	Perfect through $K = 200$, then 0% at $K = 300$.
Qwen QwQ	500 branches	50 ops/branch	0% at $K = 60$, partial recovery at $K = 100$, then 0% from $K = 150$.

Table 5: Selected failure thresholds from the updated D4, D5, and D7 batteries. The table reports the first zero-accuracy point; non-monotonic recoveries are stated explicitly rather than hidden by a single threshold.

D5 is not reducible to D4. D5 fixes the number of branches at three and varies only the operations inside each branch. This construction produces a failure profile that differs from D4. Llama 3 8B drops to 17% at 3 operations per branch and reaches 0% at 8. GPT-4o Mini and Claude 3.5 Haiku remain perfect through 3 operations but fall sharply at 5–8. Qwen3 235B and DeepSeek V3 survive the shallow levels but reach zero once branch-local composition grows. All standard models are at 0% by 50 operations per branch. Reasoning models again shift the ceiling. Qwen QwQ is non-zero through 30 operations per branch but fails at 50, 70, and 100. DeepSeek R1 0528 remains perfect through 30, drops to 75% at 50, and fails at 70 and 100.

D7 reveals counting failures that structure alone misses. D7 is the least predictable dimension from model size or structural profiles. Claude 3.5 Haiku remains at 100% from $K = 5$ through $K = 300$, while Llama 3 8B is already unstable by $K = 10$ and is zero from $K = 25$ onward. DeepSeek V3 remains strong through $K = 150$ but fails at $K = 200$ and $K = 300$. DeepSeek R1 0528 is perfect through $K = 200$ yet fails at $K = 300$, showing that extended reasoning does not guarantee exact enumeration at long repetition lengths. Qwen QwQ, GPT-4o Mini, Qwen3 235B, and Gemma 3 12B show non-monotonic recoveries. These results are consistent with counting being affected by tokenisation, context position, and output policy rather than by a smooth arithmetic difficulty curve.

D2 and D8 separate nested from sequential dependence. D2 and D8 are structurally coupled because both can create long expression spines, but they test different computations. D2 requires nested partial results to be returned through an expression tree; D8 requires a single live state to be updated repeatedly. Standard models remain strong on shallow D2 and short D8 chains, then degrade sharply as depth or step count grows. DeepSeek R1 0528 is robust on D8 through 500 steps but still degrades on extreme D2, indicating that explicit reasoning traces are more effective for linear state updates than for deeply nested return structure.

Figure 2 makes the D2/D8 contrast explicit at the level of scaling curves. In D2, most models remain at ceiling through shallow depths and then fall sharply once nested partial-result propagation becomes difficult. In D8, the x-axis is chain length rather than tree depth, and the failure pattern is different: DeepSeek R1 0528 remains at ceiling through 500 steps, while Qwen QwQ fails only at the longest levels and several standard models collapse much earlier. This matters because D2 and D8 can both create long expression spines. Their different curves show that length alone is not the mechanism. The overall nine-dimension heatmap in Figure C.1 is useful for broad model profiling, but the main analysis relies on per-dimension trajectories and thresholds because aggregate views can hide abrupt or mechanism-specific failures.

The remaining dimensions are informative but secondary in the current battery. D1 is mostly a proxy because long expressions often encode depth, counting, or sequential dependence. It remains useful for detecting prompt-length extrapolation, but it rarely identifies a unique algebraic mechanism by itself. D3 remains near ceiling through the present twelve operator levels, so future work should extend it to calculus-level and Diophantine operators.

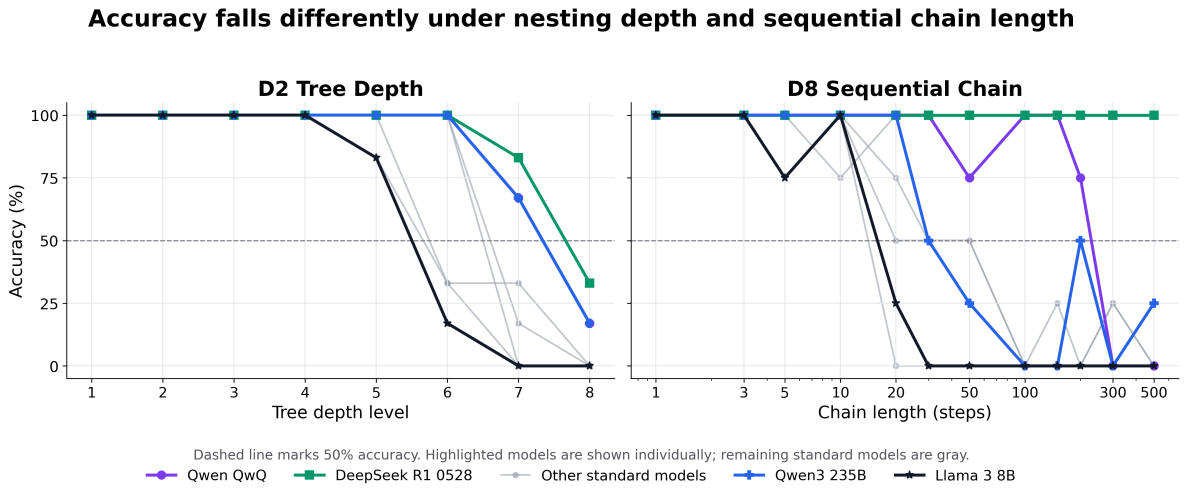


Figure 2: Level-wise accuracy curves for D2 tree depth and D8 sequential chain length. The dashed line marks 50% accuracy. DeepSeek R1 0528, Qwen QwQ, Qwen3 235B, and Llama 3 8B are highlighted individually; the remaining standard models are shown in grey. The curves show that both dimensions degrade with scale, but their trajectories differ despite related surface length. Full heatmaps, including the overall nine-dimension accuracy heatmap, are provided in Appendix C (heatmaps).

D6 is non-monotonic: quadratic factoring and cubic roots are difficult, but identity rewriting is easy despite nominal ambiguity, suggesting that familiar algebraic form matters more than the count of possible solution strategies. D9 amplifies D3 under multiplication because partial products grow quadratically with digit count (Dziri et al., 2023; Yuan et al., 2023); it is most informative when interpreted as an operator–magnitude interaction. These dimensions remain part of the full benchmark because they test mechanisms that are not captured by a single aggregate score.

Reasoning traces shift thresholds, not mechanisms. The reasoning-specialised models are strongest on D1, D4, D5, D8, and D9 because extended traces can externalise intermediate computations. However, their failures at extended D4 and D5 levels show that the generator can still identify the next threshold. This result demonstrates the importance of automatic scaling. If a model reaches ceiling at a current level, the benchmark does not need to be replaced. The same dimension definition can be extended until the failure mode reappears or is convincingly absent.

7 Conclusion

We presented a nine-dimension algebraic complexity framework and an automated CAS-verified generator that scales each dimension systematically. We also reported a cross-model evaluation over six standard instruction-tuned models and two reasoning-specialised models. The results show that aggregate accuracy hides distinct structural failures. Standard models share a D4 working-memory collapse in the 20–40 branch region, D5 exposes branch-local composition failure, D7 reveals repeated-token counting instability, and D2 and D8 separate nested from sequential dependence. Reasoning-specialised models substantially shift several thresholds, but the extended D4, D5, and D7 batteries show that the underlying limits are displaced rather than removed.

The three contributions are mutually reinforcing. The dimensions provide the explanatory vocabulary by specifying which algebraic pressure is being tested. The automatic generator provides the scaling mechanism by raising one pressure at a time while preserving the controls needed for attribution. The cross-model evaluation shows why both components are necessary. Without controlled dimensions, the shared D4 collapse, the D5/D4 separation, and the unusual D7 counting patterns would be hidden inside a single score. Conversely, without the generator, any observed threshold would be tied to a fixed dataset and would quickly become obsolete as models improve.

The broader lesson is that algebraic reasoning should be evaluated as a structured set of capacities rather than as a monolithic skill. A model may handle shallow syntax but fail nested propagation. It may solve local branches but fail horizontal storage. It may reason through long chains

but miscount repeated tokens. A useful benchmark should make these distinctions visible. The nine-dimension framework is a step toward that goal. It turns algebraic reasoning failure into a phenomenon that can be named, scaled, compared, and tested again as models change.

Limitations

The current D3 battery does not include integration, differentiation, or Diophantine equations, which are likely necessary to expose a harder operator cliff. The highest D4, D5, and D7 levels create long prompts, so future extensions should control context length and output budgets explicitly. The evaluation uses a single generated seed and temperature-zero decoding; this supports deterministic diagnosis but does not estimate variation across generated problem sets. Non-monotonic points in D4, D5, D7, and D8 should therefore be re-run with additional seeds before being interpreted as robust recovery. Finally, the working-memory interpretation is supported by controlled curves and prior theory, but this paper does not directly intervene on model internals.

Acknowledgements

We thank the reviewers for their constructive feedback and our colleagues for discussions during the development of this evaluation framework. The authors acknowledge the use of AI-assisted writing tools, including Claude and Gemini, solely to improve clarity, grammar, and presentation. All experimental designs, generated problems, model evaluations, analyses, findings, interpretations, and conclusions are the authors' own responsibility.

References

Alibaba Cloud. 2025. QwQ: A thinking model by Qwen. Technical report, Alibaba Cloud.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 38546–38556. Curran Associates, Inc.

Yingshan Chang and Yonatan Bisk. 2024. Language models need inductive biases to count inductively. *arXiv preprint arXiv:2405.20131*.

Shubh Chapra, Dhruv Kumar, Murari Mandal, and Yash Sinha. 2026. The complexity ceiling benchmark: A multi-domain evaluation of sequential reasoning under depth scaling. In *ICML 2026 Workshop on Combining Theory and Benchmarks*.

Sheng Chen, Chao Tian, Bo Hu, Kang Chen, Zheng Liu, Zheng Zhang, and Ji Zhou. 2025. Arrows of math reasoning data synthesis for large language models: Diversity, complexity and correctness. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 4665–4669.

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

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. Technical report, DeepSeek-AI.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems*, volume 36.

Jingxuan Fan, Sarah Martinson, Erik Y. Wang, Kaylie Hausknecht, Jonah Brenner, Danxian Liu, Nianli Peng, Corey Wang, and Michael P. Brenner. 2024. HARDMath: A benchmark dataset for challenging problems in applied mathematics. *arXiv preprint arXiv:2410.09988*.

Dongyu Gong and Hantao Zhang. 2024. Self-attention limits working memory capacity of transformer-based models. *arXiv preprint arXiv:2409.10715*.

Aryan Gulati, Brando Miranda, Eric Chen, Emily Xia, Kai Fronsdal, Bruno De Moraes Dumont, and Sanmi Koyejo. 2025. Putnam-AXIOM: A functional & static benchmark for measuring higher level mathematical reasoning in LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 20723–20747. PMLR.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*.

Arian Hosseini, Alessandro Sordoni, Daniel Toyama, Aaron Courville, and Rishabh Agarwal. 2024. Not all LLM reasoners are created equal. *arXiv preprint arXiv:2410.01748*.

Kanav Kapoor, Dhruv Kumar, Jagat Sesh Challa, Murari Mandal, and Yash Sinha. 2026. GraphStateEval: A step-by-step evaluation framework for graph algorithm execution in large language models via intermediate state tracing. In *ICML 2026 Workshop on Combining Theory and Benchmarks*.

Guillaume Lample and François Charton. 2020. Deep learning for symbolic mathematics. In *International Conference on Learning Representations (ICLR)*.

Chaitanya Manem, Pratik Prabhanjan Brahma, Prakamya Mishra, Zicheng Liu, and Emad Barsoum. 2025. SAND-Math: Using LLMs to generate novel, difficult and useful mathematics questions and answers. In *NeurIPS 2025 Workshop on MATH-AI*.

Larisa Markeeva, Sean McLeish, Borja Ibarz, Wilfried Bounsi, Olga Kozlova, Alex Vitvitskyi, Charles Blundell, Tom Goldstein, Avi Schwarzschild, and Petar Veličković. 2024. The CLRS-Text algorithmic reasoning language benchmark. *arXiv preprint arXiv:2406.04229*.

William Merrill and Ashish Sabharwal. 2023. The expressive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2025. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations*.

Mohammad Ramezanali, Mo Vazifeh, and Paolo Santi. 2025. seqBench: A tunable benchmark to quantify sequential reasoning limits of LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33771–33792, Suzhou, China. Association for Computational Linguistics.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations (ICLR)*.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *arXiv preprint arXiv:2506.06941*.

Aaditya K. Singh and DJ Strouse. 2024. Tokenization counts: the impact of tokenization on arithmetic in frontier LLMs. *arXiv preprint arXiv:2402.14903*.

Rishabh Singh, Sumit Gulwani, and Sriram K. Rajamani. 2012. Automatically generating algebra problems. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1620–1628.

Peiyang Song, Pengrui Han, and Noah Goodman. 2026. Large language model reasoning failures. *arXiv preprint arXiv:2602.06176*.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. 2024. MathScale: Scaling instruction tuning for mathematical reasoning. *arXiv preprint arXiv:2403.02884*.

Yi Xu, Roger Smeets, and Rafael Bidarra. 2021. Procedural generation of problems for elementary math education. *International Journal of Serious Games*, 8(2).

Gilad Yehudai, Haim Kaplan, Guy Dar, Royi Rassin, Asma Ghandeharioun, Mor Geva, and Amir Globerson. 2024. When can transformers count to n? *arXiv preprint arXiv:2407.15160*.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. 2023. How well do large language models perform in arithmetic tasks? *arXiv preprint arXiv:2304.02015*.

Xiang Zhang, Juntai Cao, and Chenyu You. 2024. Counting ability of large language models and impact of tokenization. *arXiv preprint arXiv:2410.19730*.

Jun Zhao, Jingqi Tong, Yurong Mou, Ming Zhang, Qi Zhang, and Xuanjing Huang. 2024. Exploring the compositional deficiency of large language models in mathematical reasoning through trap problems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 16361–16376.

Appendix Roadmap

Section	Contents
A. Benchmark Specification	Dimension definitions, fixed controls, level schedules, and example generated items.
B. Evaluation and Reproducibility	Models, prompt template, metrics, and evaluation rules.
C. Detailed Results	Overall accuracy summary and dimension-specific tables and heatmaps.
D. Cross-Dimensional Couplings	Directional interaction matrix and generator-level explanations.

A Benchmark Specification

The examples below are illustrative infix renderings of the generator families. The actual benchmark samples 50 CAS-verified instances per level using seed 42. Ellipses indicate repeated generator structure rather than omitted benchmark evidence.

A.1 D1: Syntactic length

D1 measures the prefix-token count of the generated expression. The active failure mechanism is length extrapolation: a model may know the local arithmetic pattern but fail when the same pattern is stretched beyond familiar sequence lengths (Anil et al., 2022; Markeeva et al., 2024). The D1 battery therefore uses simple right-growing addition chains with single-digit operands, ranging from 5 to 751 tokens.

Active variable	Prefix token count
Fixed controls	Single-digit operands, addition chains, no branching beyond the chain template.
Level schedule	5, 11, 21, 51, 75, 101, 201, 501, 751 tokens.
Low level	`3+2+6`
Middle level	`5+8+1+7+4+…`
High level	`2+3+4+9+1+…` with hundreds of prefix tokens.

A.2 D2: Tree depth

D2 measures the longest root-to-leaf path in the expression tree. Increasing depth raises the amount of nested partial-result propagation required before the root can resolve. The isolated battery controls width, operand size, and operator difficulty so that the active pressure is vertical nesting rather than hard arithmetic. This follows prior evidence that neural models lose accuracy as nesting depth increases (Saxton et al., 2019; Dziri et al., 2023).

Active variable	Maximum tree depth
Fixed controls	Simple operators, small operands, no wide branch accumulation.
Level schedule	Depth 1–8 for standard models; extended to 10, 20, and 50 for reasoning models.
Low level	`3+2`
Middle level	`3+7*(6+5)`
High level	`3+7*(6+5*(2+5*(3+3*4)))`.

A.3 D3: Operator hardness

D3 measures the identity and intrinsic difficulty of the operator under test. The current battery orders twelve operator levels from low-hardness arithmetic through roots, exponentials, trigonometric operators, and powers while holding structural dimensions at minimum values. The present results remain near ceiling, which suggests that future D3 extensions should include calculus-level and Diophantine operators.

Active variable	Operator identity
Fixed controls	Depth, branch count, operand size, counting load, and chain length fixed.
Level schedule	Twelve ordered levels from negation through `pow-hard`.
Low level	`-7`
Middle level	`sqrt(49)`
High level	`pow(7,5)` or trigonometric forms.

A.4 D4: Working-memory load

D4 measures the number of independent intermediate results that must coexist before combination. Each problem is a sum of K independent single-digit products. The products are arithmetically simple. The intended challenge is to compute many branch results, retain them, and sum them correctly. This targets the absence of a register-like working memory in standard self-attention (Gong and Zhang, 2024) and connects to thresholded reasoning-collapse results (Shojaee et al., 2025).

Active variable	Number of parallel branches K
Fixed controls	Each branch is a trivial single-digit product.
Level schedule	2, 4, 8, 12, 20, 25, 30, 40, 50, 60, 75, 100, 150, 200, 500, 1000, 3000.
Low level	`3*2+6*5`
Middle level	`3*2+6*5+7*4+…`
High level	A sum of hundreds or thousands of independent products.

A.5 D5: Compositional branching

D5 measures the number of operations inside each branch before that branch resolves to a scalar. The generator fixes three branches joined by two root operators, then increases branch-internal operations from 0 to 100. D5 is separated from D4 by construction. D4 varies how many simple branch outputs must be stored, whereas D5 varies how difficult each branch is before storage (Zhao et al., 2024; Hosseini et al., 2024).

Active variable	Operations per branch
Fixed controls	Exactly three branches joined by two root operators.
Level schedule	0, 1, 2, 3, 5, 8, 12, 20, 30, 50, 70, 100.
Low level	`3+2+6`
Middle level	`(3+2*4)+(8-1*5)+(6+7)`
High level	Three fixed branches, each with tens of internal operations.

A.6 D6: Solution ambiguity

D6 measures the number of structurally distinct valid solution strategies. A linear equation such as `3x+6=-9` has a nearly canonical path because isolating x is the dominant procedure. A quadratic such as `x^2-5x+6=0` can be solved by factoring, by applying the quadratic formula, or by testing candidate roots, and these routes expose different intermediate representations. A rational identity such as `(a^2-b^2)/(a+b)` can be approached by expansion, cancellation after recognising a difference of squares, or substitution-based checking. These examples are not difficult because the final arithmetic is large. They are difficult because the model must select a useful algebraic form before computation becomes straightforward.

The D6 battery therefore treats ambiguity as a strategy-selection burden rather than as a count of final answers. Root-extraction, factoring, rational simplification, and identity-rewrite items can all have compact solutions, but the compact solution is available only after the expression is represented in the right form. This connects D6 to symbolic-mathematics settings where search, form recognition, and route selection interact with model decoding behaviour (Lample and Charton, 2020).

Active variable	Problem type
Fixed controls	Numeric size and expression structure kept small inside each type.
Level schedule	Eight types from linear equations to identity rewrites.
Low level	`3x+6=-9`
Middle level	`x^2-5x+6=0`
High level	`(8^2-2^2)/(8+2)` or identity-rewrite instances.

A.7 D7: Counting load

D7 measures the repeated-term count K for identical operands. The repeated value and operation are intentionally simple, and paired controls remove the counting demand. Failures are therefore attributed to enumeration and cardinality tracking rather than final arithmetic. This dimension is grounded in work on counting, tokenisation, positional encodings, and context effects (Chang and Bisk, 2024; Zhang et al., 2024; Yehudai et al., 2024).

Active variable	Repeated-term count K
Fixed controls	Repeated operand value and arithmetic operation fixed.
Level schedule	$K = 5, 7, 10, 12, 18, 25, 60, 100, 150, 200, 300$.
Low level	`7+7+7+7+7`
Middle level	7 repeated 60 times
High level	7 repeated 300 times.

A.8 D8: Sequential chain length

D8 measures the longest strictly dependent step path. Each intermediate value becomes the input to the next operation, so an early error contaminates all later steps. Unlike D4, only one live state must be carried forward. This isolates cascading sequential error (Merrill and Sabharwal, 2023; Ramezanali et al., 2025).

Active variable	Dependent chain length
Fixed controls	One live intermediate value, small operands, no branch accumulation.
Level schedule	1, 3, 5, 10, 20, 30, 50, 100, 150, 200, 300, 500 steps.
Low level	`start=4; +3`
Middle level	`(((4+3)*2)-5)+…`
High level	A 500-step dependent update chain.

A.9 D9: Numeric magnitude

D9 measures the maximum digit count in multiplication operands. The generator fixes the operator to multiplication and increases operand size from 1 to 15 digits. This tests partial-product accumulation: multiplication creates many cross-digit interactions, unlike addition-like carry propagation (Dziri et al., 2023; Yuan et al., 2023). Tokenisation can further change arithmetic behaviour for equivalent numeric values (Singh and Strouse, 2024).

Active variable	Operand digit count
Fixed controls	Operator fixed to multiplication; minimal depth and branch count.
Level schedule	1, 2, 4, 6, 8, and 15 digits.
Low level	`3*2`
Middle level	`4829*7316`
High level	A 15-digit by 15-digit multiplication.

B Evaluation and Reproducibility

B.1 Models

Model	Params	Type	Architecture	Role in Evaluation
GPT-4o Mini	Undisclosed	Standard	Undisclosed	Frontier small instruction model.
Claude 3.5 Haiku	Undisclosed	Standard	Undisclosed	Frontier small instruction model.
Qwen3 235B	235B total 22B active	Standard	MoE decoder	Largest standard model in the evaluation.
DeepSeek V3	671B total 37B active	Standard	MoE decoder	Large mixture-of-experts standard model.
Gemma 3 12B	12B	Standard	Decoder-only	Mid-size open-weight instruction model.
Llama 3 8B	8B	Standard	Decoder-only	Small open-weight baseline.
DeepSeek R1 0528	671B total 37B active	Reasoning	MoE + CoT	Extended reasoning trace model.
Qwen QwQ	32.5B	Reasoning	Decoder + CoT	Extended reasoning trace model.

Table B.1: The eight evaluated models. Proprietary hosted model sizes are not publicly disclosed. For mixture-of-experts models, total and active denote total parameters and parameters active per forward pass.

B.2 Prompt Template

All models received the same instruction template at temperature zero:

> Solve the following algebraic expression or equation step by step. Give the final result on a separate line using exactly the format `ANSWER: value`. Do not include any other text after the answer line.

The scorer extracts the final answer line, normalises simple numeric formatting, and compares it to the CAS result. A response with no parseable final answer is counted as a model failure. No model is given tool access, self-consistency sampling, majority voting, or human correction.

B.3 Metrics and Error Accounting

Metric	Definition
Accuracy	Percentage of valid responses within CAS tolerance.
Failure threshold	First level below 50%; later recoveries are reported separately.
First zero-accuracy level	First evaluated level at 0% accuracy, as reported in Table 5.
Failure pattern	Qualitative curve type: collapse, model-discriminating, or non-monotonic.
Error accounting	API and CAS errors are excluded from accuracy denominators.

Table B.2: Evaluation metrics and error-accounting rules.

C Detailed Results

In these heatmaps, rows correspond to models and columns correspond either to dimensions or to increasing levels within one dimension. Darker green indicates higher accuracy, and red indicates collapse. The captions identify the interpretation most relevant to the main paper.

C.1 Overall Accuracy

Dim.	Dimension	GPT-4o	Claude	Qwen3	DS V3	Gemma	Llama	R1	QwQ
D1	Syntactic Length	60.78%	58.89%	73.33%	65.22%	35.56%	48.22%	100%	82.22%
D2	Tree Depth	75.00%	68.62%	85.50%	77.12%	66.62%	62.50%	89.50%	85.50%
D3	Operator Score	100.00%	100.00%	97.92%	100.00%	97.92%	95.83%	100.00%	100.00%
D4	Working Memory	27.94%	27.94%	29.88%	42.59%	23.53%	32.35%	94.12%	80.88%
D5	Comp. Branching	42.25%	42.25%	45.00%	46.67%	39.17%	26.33%	81.25%	68.33%
D6	Sol. Ambiguity	71.43%	76.14%	88.00%	78.57%	78.43%	48.00%	81.62%	83.38%
D7	Counting Load	61.82%	100%	56.82%	77.73%	54.55%	21.82%	90.91%	61.36%
D8	Seq. Chain Length	33.33%	47.92%	54.17%	52.08%	45.83%	33.33%	100%	79.17%
D9	Numeric Magnitude	70.83%	79.17%	62.50%	62.33%	41.67%	43.67%	100%	87.50%

Table C.1: Per-dimension mean accuracy summary across all nine dimensions.

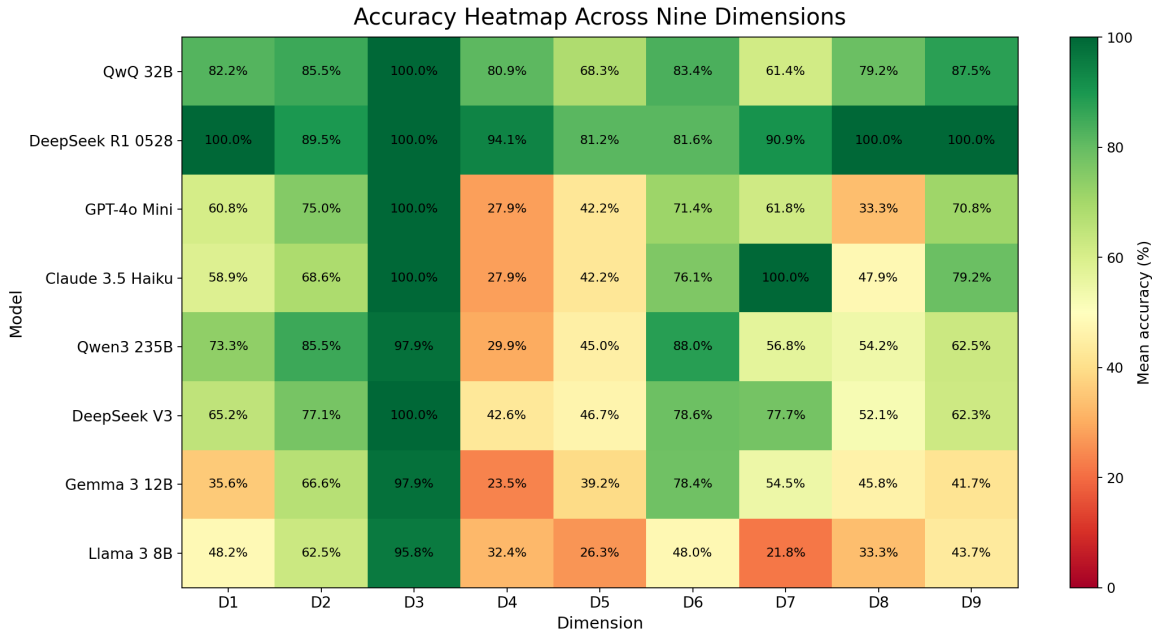


Figure C.1: Overall nine-dimension mean-accuracy heatmap. This appendix view is useful for broad model profiling, but it should be read alongside the per-dimension curves and heatmaps because mean accuracy can hide abrupt threshold failures on individual dimensions.

C.2 D4: Working-memory load

Model	12	20	25	30	40	50	200	500	1000	3000	Type
GPT-4o Mini	100%	75%	0%	0%	0%	0%	0%	0%	0%	0%	Standard
Claude 3.5 Haiku	100%	50%	0%	0%	25%	0%	0%	0%	0%	0%	Standard
Qwen3 235B	100%	25%	25%	25%	0%	33%	0%	0%	0%	0%	Standard
DeepSeek V3	100%	100%	75%	0%	50%	33%	0%	0%	0%	0%	Standard
Gemma 3 12B	75%	25%	0%	0%	0%	0%	0%	0%	0%	0%	Standard
Llama 3 8B	75%	75%	100%	0%	0%	0%	0%	0%	0%	0%	Standard
DeepSeek R1 0528	100%	100%	100%	100%	100%	100%	100%	100%	100%	0%	Reasoning
Qwen QwQ	100%	100%	100%	100%	75%	100%	100%	0%	0%	0%	Reasoning

Table C.2: D4 working-memory accuracy at selected branch counts. Columns denote parallel branches.

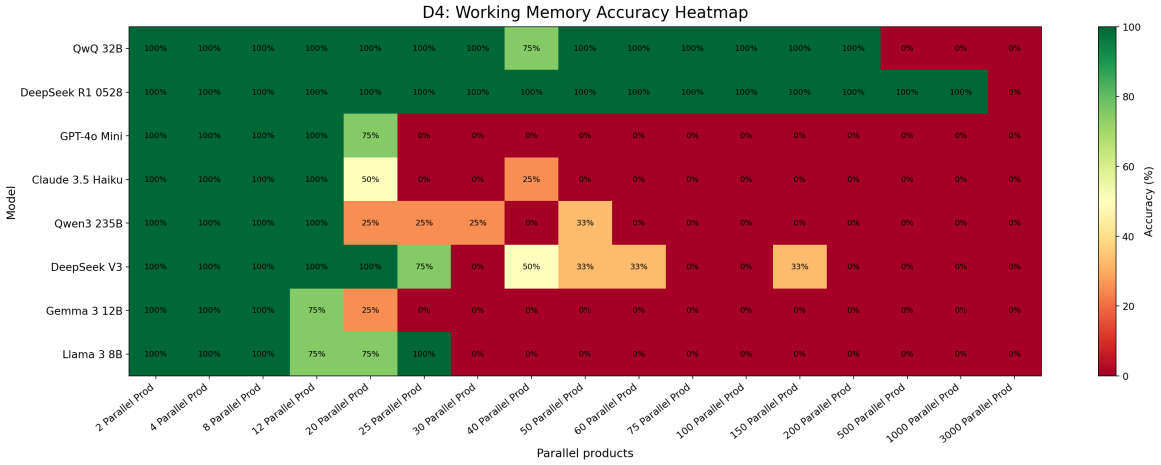


Figure C.2: D4, working-memory load. Standard models remain strong at small branch counts but enter a failure band from roughly 20 to 40 branches; every standard model reaches at least one zero-accuracy point by 40 branches. Qwen QwQ fails at 500, 1000, and 3000 branches, while DeepSeek R1 0528 fails at 3000.

C.3 D5: Compositional branching

Model	0	1	2	3	5	8	12	20	30	50	70	100
GPT-4o Mini	100%	100%	100%	100%	67%	40%	0%	0%	0%	0%	0%	0%
Claude 3.5 Haiku	100%	100%	100%	100%	67%	20%	20%	0%	0%	0%	0%	0%
Qwen3 235B	100%	100%	100%	100%	100%	20%	0%	0%	20%	0%	0%	0%
DeepSeek V3	100%	100%	100%	100%	100%	40%	20%	0%	0%	0%	0%	0%
Gemma 3 12B	100%	100%	100%	83%	67%	20%	0%	0%	0%	0%	0%	0%
Llama 3 8B	100%	83%	83%	17%	33%	0%	0%	0%	0%	0%	0%	0%
DeepSeek R1 0528	100%	100%	100%	100%	100%	100%	100%	100%	100%	75%	0%	0%
Qwen QwQ	100%	100%	100%	100%	100%	80%	100%	80%	60%	0%	0%	0%

Table C.3: D5 branch-complexity accuracy by operations per branch.

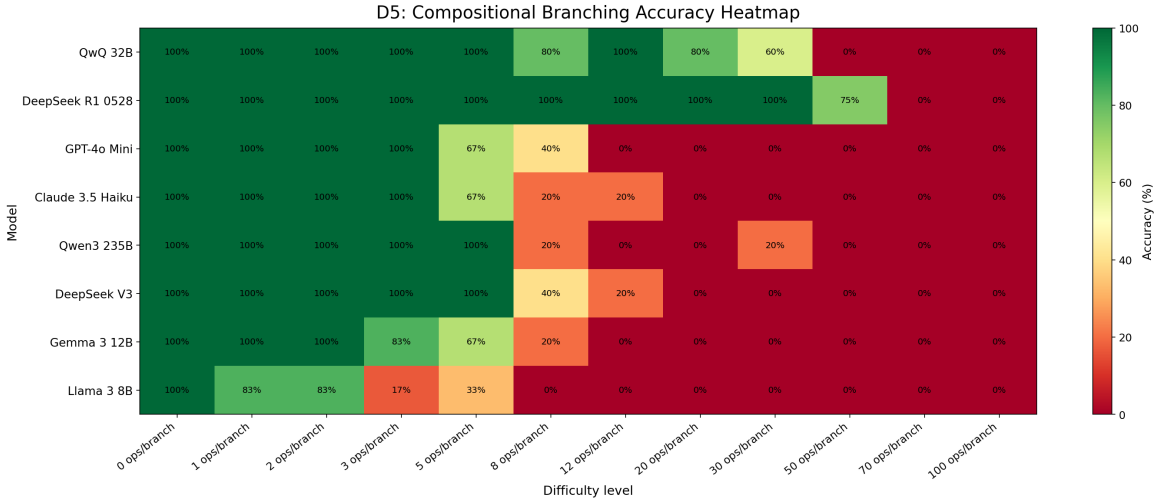


Figure C.3: D5, branch sub-expression complexity. The number of branches is fixed at three; only the operations inside each branch increase. All standard models reach 0% by 50 operations per branch. Qwen QwQ fails at 50, 70, and 100, while DeepSeek R1 0528 fails at 70 and 100.

C.4 D7: Counting load

Model	5	7	10	12	18	25	60	100	150	200	300
GPT-4o Mini	100%	100%	100%	100%	100%	80%	0%	100%	0%	0%	0%
Claude 3.5 Haiku	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Qwen3 235B	100%	100%	100%	100%	100%	100%	0%	25%	0%	0%	0%
DeepSeek V3	100%	100%	100%	80%	100%	100%	100%	100%	75%	0%	0%
Gemma 3 12B	100%	100%	100%	100%	100%	0%	100%	0%	0%	0%	0%
Llama 3 8B	80%	100%	20%	20%	20%	0%	0%	0%	0%	0%	0%
DeepSeek R1 0528	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	0%
Qwen QwQ	100%	100%	100%	100%	100%	100%	0%	75%	0%	0%	0%

Table C.4: D7 counting-load accuracy by repeated-term count K.

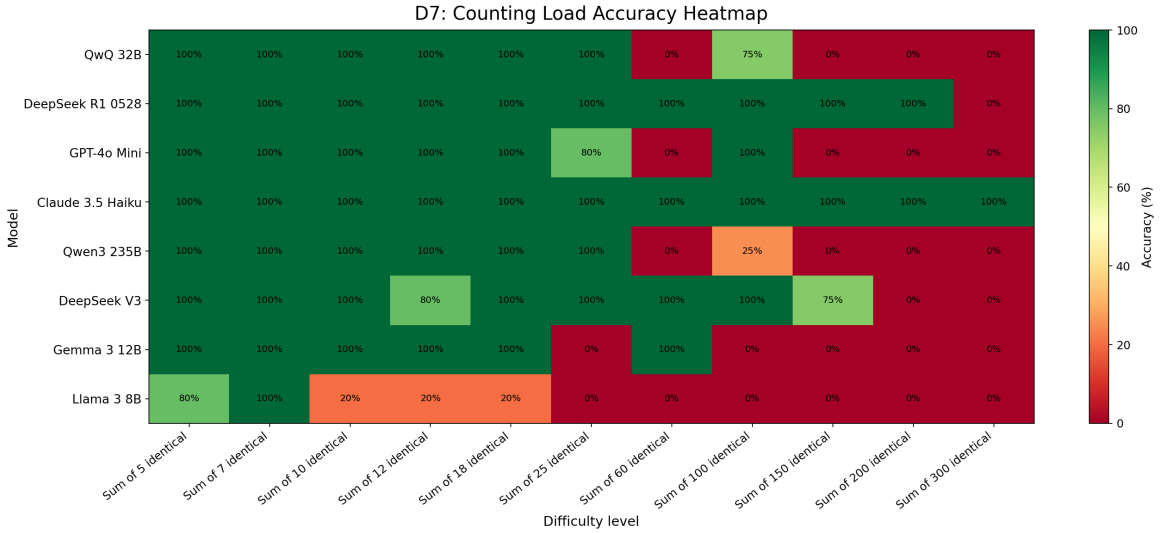


Figure C.4: D7, counting load. Claude 3.5 Haiku stays at 100% through $K = 300$. DeepSeek R1 0528 remains perfect through $K = 200$ but fails at $K = 300$. Several other models show non-monotonic recoveries, confirming that counting is not a smooth function of expression length alone.

C.5 D8: Sequential chain length

Model	1	5	10	20	30	50	100	500
GPT-4o Mini	100%	100%	100%	0%	0%	0%	0%	0%
Claude 3.5 Haiku	100%	100%	100%	50%	50%	50%	0%	0%
Qwen3 235B	100%	100%	100%	100%	50%	25%	0%	25%
DeepSeek V3	100%	100%	100%	75%	50%	50%	0%	0%
Gemma 3 12B	100%	100%	75%	100%	50%	25%	0%	0%
Llama 3 8B	100%	75%	100%	25%	0%	0%	0%	0%
DeepSeek R1 0528	100%	100%	100%	100%	100%	100%	100%	100%
Qwen QwQ	100%	100%	100%	100%	100%	75%	100%	0%

Table C.5: D8 sequential-chain accuracy at selected step counts.

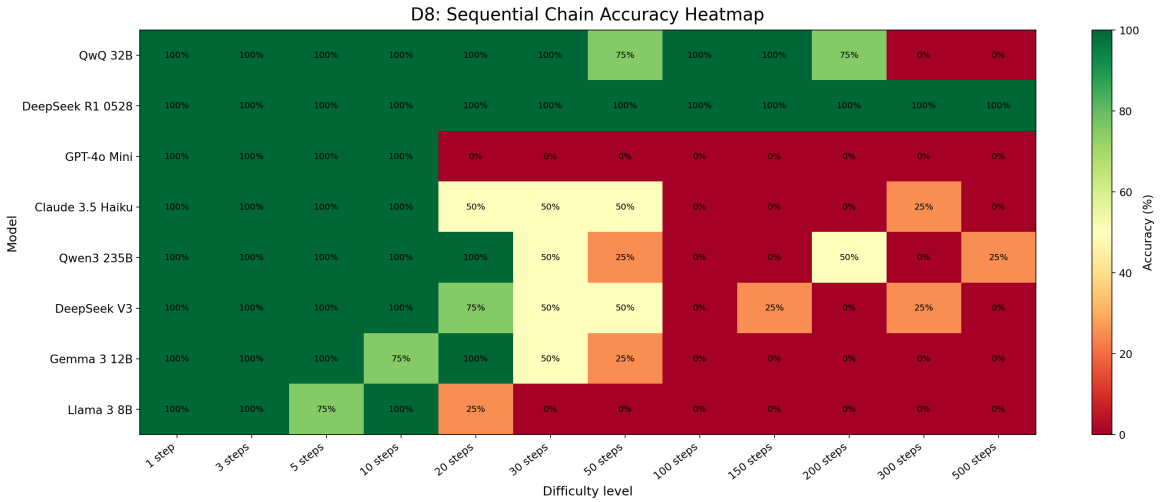


Figure C.5: D8, sequential chain length. Each added step depends on the previous state, so errors cascade through the chain. DeepSeek R1 0528 remains at ceiling, while Qwen QwQ eventually fails at the longest levels.

C.6 D1: Syntactic length

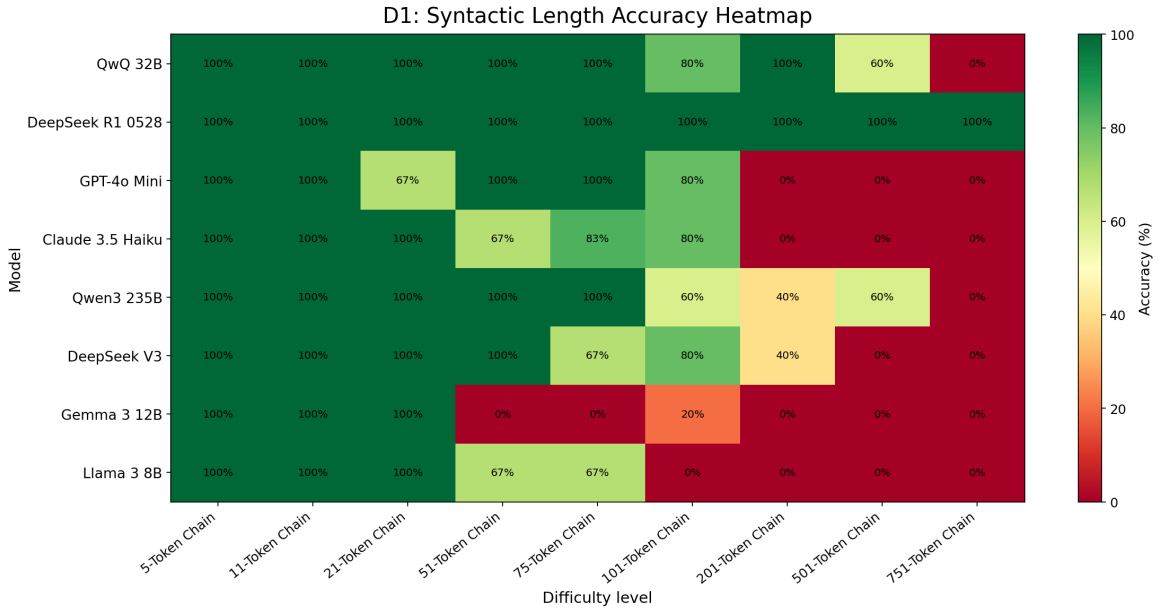


Figure C.6: D1, syntactic length. Long simple chains expose length extrapolation, but D1 is best read as a proxy because long expressions often arise from depth, counting, or sequential structure.

C.7 D2: Tree depth

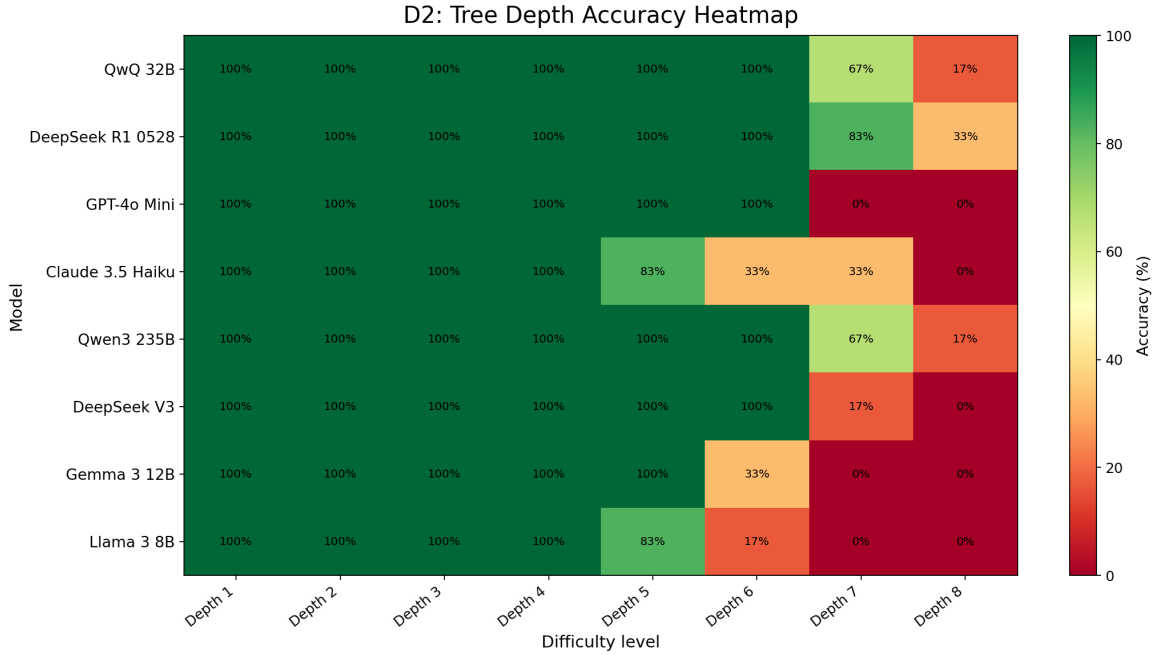


Figure C.7: D2, nesting depth. Accuracy is stable at shallow depths, separates around depths 5–6, and collapses for most standard models by depths 7–8.

C.8 D3: Operator hardness

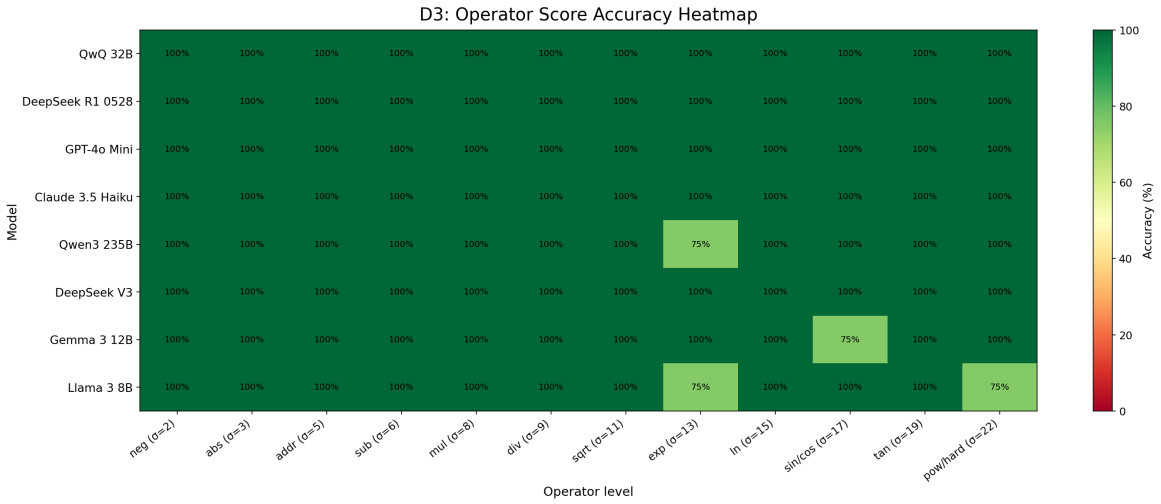


Figure C.8: D3, operator score. Most models remain near ceiling through the current operator ladder, indicating that the next operator cliff likely lies beyond the present battery.

C.9 D6: Solution ambiguity

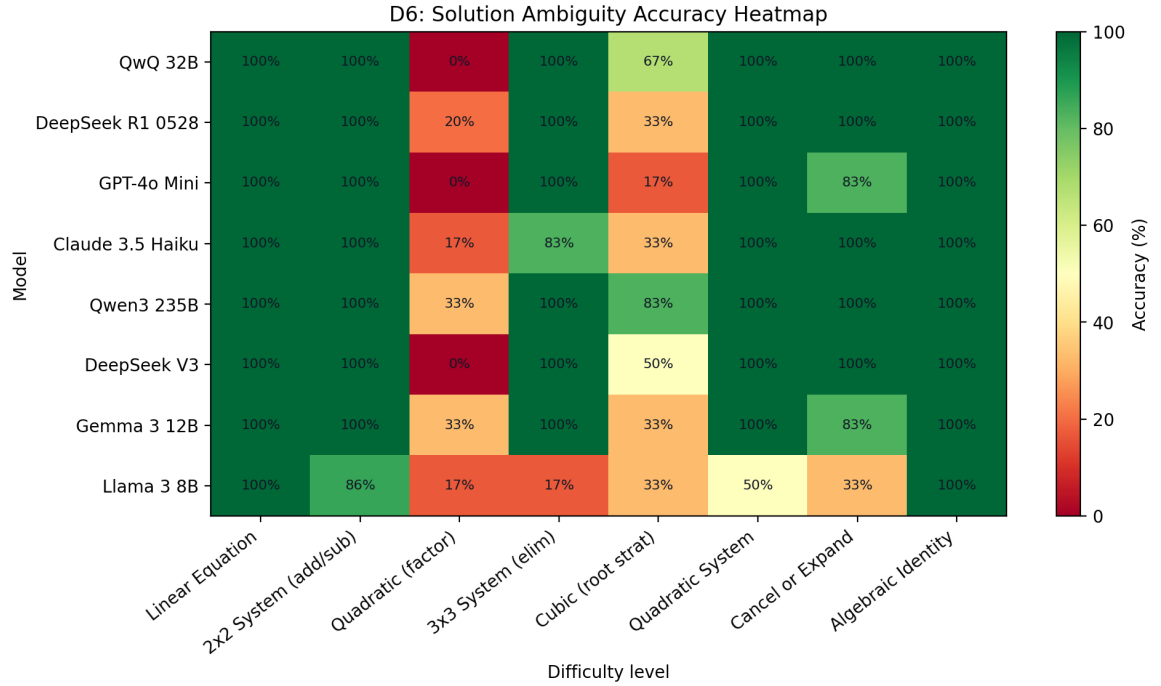


Figure C.9: D6, solution ambiguity. The pattern is non-monotonic: quadratic factoring and cubic-root forms are difficult, while identity rewriting is easy for all models.

C.10 D9: Numeric magnitude

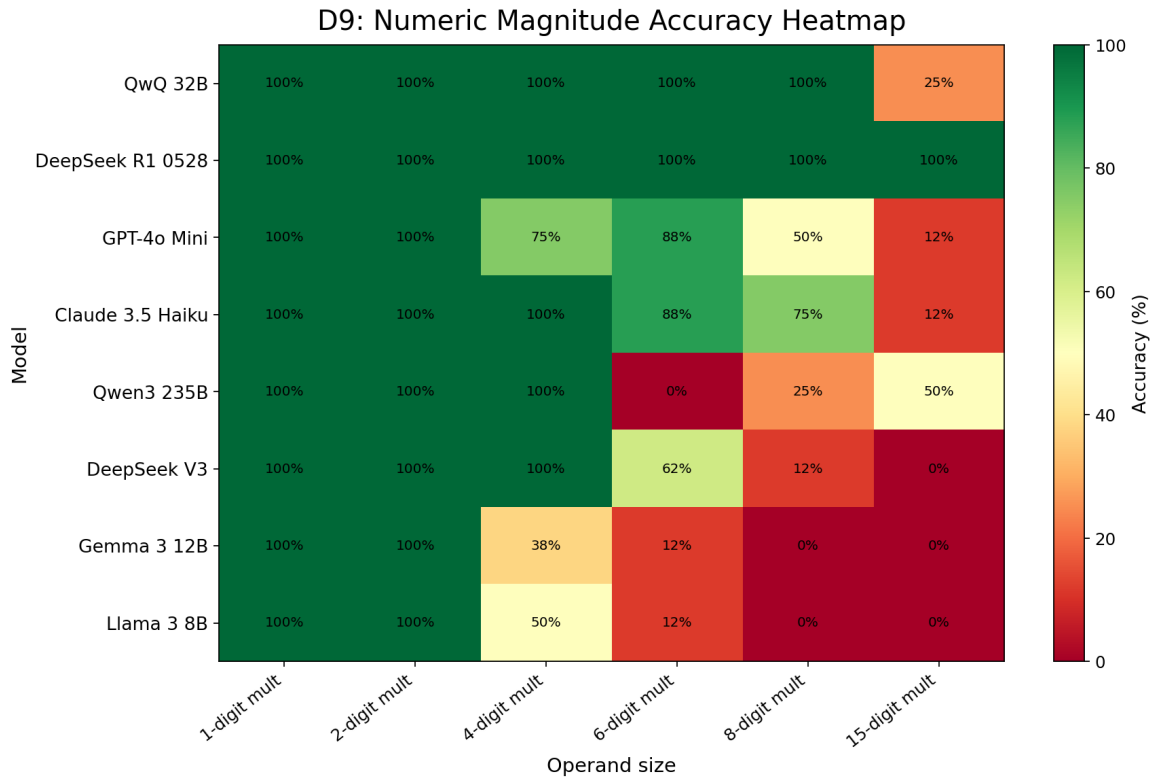


Figure C.10: D9, numeric magnitude. Large operands become difficult mainly under multiplication because the number of partial products grows quadratically with digit count.

D Cross-Dimensional Couplings

Generator-level isolation means one control variable is intentionally varied, but it does not imply that all measured surfaces remain fixed. Table D.1 reports directional co-movement. **S** means structurally guaranteed, **M** means template-induced or high-probability, **W** means weak finite-sample bleed, and **N** means fixed by construction.

Increased ↓ / Affected →	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**
D1 Syntactic Length	—	W	N	N	N	N	**S**	M	N
D2 Tree Depth	M	—	W	W	N	N	N	**S**	N
D3 Operator Score	W	N	—	N	N	N	N	N	M
D4 Working Memory	**S**	**S**	N	—	N	N	W	N	N
D5 Branch Complexity	**S**	**S**	N	N	—	N	N	M	N
D6 Solution Ambiguity	W	N	M	N	N	—	N	N	W
D7 Counting Load	**S**	M	N	N	N	N	—	W	N
D8 Sequential Chain	**S**	**S**	W	N	N	N	N	—	N
D9 Numeric Magnitude	W	N	M	N	N	N	N	N	—

Table D.1: Directional cross-dimensional interaction matrix. Entry (i, j) records how increasing D_i affects D_j.

D.1 D4 to D1 and D2

The D4 generator constructs a right-nested sum of N independent products of the form `* a b`. Each branch contributes three prefix tokens and each joining + contributes one operator, giving $4N - 1$ tokens. At $N = 200$, the expression has 799 tokens; at $N = 3000$, it has 11,999 tokens. The right-nested construction also increases tree depth, so D4-to-D1 and D4-to-D2 couplings are structurally guaranteed.

D.2 D5 to D1, D2, and D8

D5 fixes three branches and increases branch-internal operations. If `sub_ops` denotes operations per branch, token count grows approximately as $3(2 \cdot \texttt{sub\_ops} + 1) + 2$; at 100 operations per branch, this is about 605 prefix tokens. The maximum tree depth grows with branch-internal depth, and a right-spine branch creates a local sequential dependency path, explaining the moderate D5-to-D8 coupling.

D.3 D7 and D8 to Length and Depth

D7 produces a repeated sum with K operands and $K - 1$ joining operators, so the prefix string has $2K - 1$ tokens. When right-nested, it also creates a long expression spine, but the intended mechanism remains enumeration. D8 appends one operator–operand pair per dependent step, giving $2 \cdot \texttt{steps} + 1$ tokens and a deep expression spine. For this reason, D2 and D8 should be interpreted jointly without being collapsed into one dimension.

Same Text, Different Transitions: Action Insufficiency in Lean Provers

Edison Yang[1] **Neel Marripalapu**[2] **Akshaj Agadi**[3] **Andrew Kim**[1]
[1]Lambert High School [2]The University of Texas at Dallas [3]Texas A&M University
edisonly2200@gmail.com

Abstract

Neural Lean provers usually give the model only the printed text of a proof state, and some searches reuse it to decide when two states are the same. Both uses assume that states printing identically behave identically under the same tactic. We show that this assumption fails. Searching Mathlib, we find 70 verified classes of identically printed states; in 68 of them, one tactic succeeds on one member and fails on another, yielding 103 contradictory cases verified by Lean. Controlled experiments trace the difference to elaborated structure that Lean's notation hides, and reproduce it on current Lean and Mathlib. A compact structural representation recovers the missing signal for mechanisms seen in training and fails to transfer to unseen ones. In a constructed benchmark with a known proof, merging states by identical text discards that proof in 48.4% of alias arrivals, while a structural identity prevents every loss at 8% runtime overhead. This rate is conditional on meeting an alias; in blind sampling, confirmed aliases arise at only 0.29% of search roots. Text written for people is thus not necessarily a safe identifier for machine state, and we suggest giving the model that text while identifying search states by structure.

1 Introduction

Neural Lean provers are given a textual rendering and asked to select a tactic. A pretty-printer is not injective, and by itself that is unremarkable: notation exists to suppress detail. The consequential question is whether the suppressed detail changes what a tactic does, because the same text is used as the policy's input and, in some systems, as the identity of a search node. We show that this interface can map action-distinct states to the same text. Table 1 is one instance: changing only the order of `intro h` and `simp` reaches a common 206-byte rendering, after which another `simp` fails on one state and succeeds on the other. The outcomes repeat on fresh sessions.

	history	then `simp`
s_1	`intro h; simp`	fails (×3)
s_2	`simp; intro h`	succeeds (×3)

Table 1: Two states in one Mathlib proof (`lintegral_ofReal_le_lintegral_nnnorm`), reached from a common ancestor by different tactic sequences. Both render to an identical 206-byte string; the same tactic, replayed three times per state on fresh sessions, deterministically fails on one and succeeds on the other. The hidden difference is the dominant coercion mechanism of §5: the two shallow explicit renderings are 1,167 and 1,183 bytes and carry `ENNReal.ofNNReal h` against `@WithTop.some.{0} NNReal h` in two positions, with equal environment digests and distinct structural digests.

Learned tactic generators for interactive theorem provers now close a substantial share of held-out library and competition problems (Bansal et al., 2019; Polu and Sutskever, 2020; Han et al., 2022; Lample et al., 2022; Yang et al., 2023; Jiang et al., 2023), on benchmarks such as miniF2F (Zheng et al., 2022) and PutnamBench (Tsoukalas et al., 2024). Many share one interface decision: the policy conditions on a printed rendering of the proof state, and some best-first search implementations also use that text as the identity of a search node (Yang et al., 2023; Lample et al., 2022). These uses assume that the observation identifies the state's response to an action, a transition-preservation condition required by safe state abstraction (Givan et al., 2003; Li et al., 2006). It is also implicit in learning tactic semantics from observed transitions (Samoylov and Vosoughi, 2025; Lamont et al., 2025). We test that assumption directly.

The issue also has a supervised-learning formulation. When the state rendering and tactic text are identical but compiler acceptance differs, an acceptance predictor given only text receives con-

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 240–253
October 29, 2026 ©2026 Association for Computational Linguistics

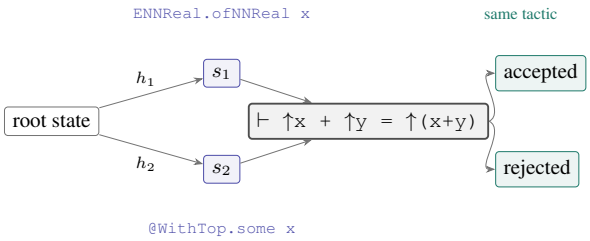


Figure 1: Observation aliasing. The point of interest lies in the two resulting states, which are indistinguishable to the policy; tactic order affecting an outcome is expected on its own. They differ only in elaborated structure the printer collapses, so the rendering, and therefore the entire input supplied to the evaluated state-only generators, is byte-identical, yet the same tactic is accepted on one and rejected on the other. The histories serve only to reach the two states; what matters is the pair of outcomes at a single observation. The goal shown is the compact constructed pair of §5, chosen for legibility; Table 1 is a natural instance of the same mechanism, whose rendering runs to 206 bytes.

tradictory deterministic labels. The resulting error cannot be removed by model scale or optimization while the observation is fixed. This affects tactic-effect prediction, reranking, transition modeling, imitation learning, dataset construction, and evaluation wherever the rendered state serves as a complete conditioning variable.

The hidden difference in Figure 1 lies in how each coercion is elaborated: `ENNReal.ofNNReal x` versus `@WithTop.some.{0} NNReal x`. They are definitionally equal and both print as `↑x`, but `simp`'s syntactic progress test distinguishes them. We call two such states *observation aliases*: this is state, or perceptual, aliasing (Whitehead and Ballard, 1991; Singh et al., 1994) instantiated in a deterministic and compiler-checkable setting, where the compiler verifies both sides of a contradictory label pair. Figure 2 organizes the evidence by logical implication and keeps incidence and consequence on their respective denominators. Its arrows ask distinct questions: whether aliases exist, whether the hidden difference changes an action, whether structural information explains and predicts the change, and whether an unsafe retention rule can discard a proof. Evidence for one arrow does not substitute for the denominator of another.

Over Mathlib (The mathlib Community, 2020) in Lean 4 (Moura and Ullrich, 2021) through Lean-

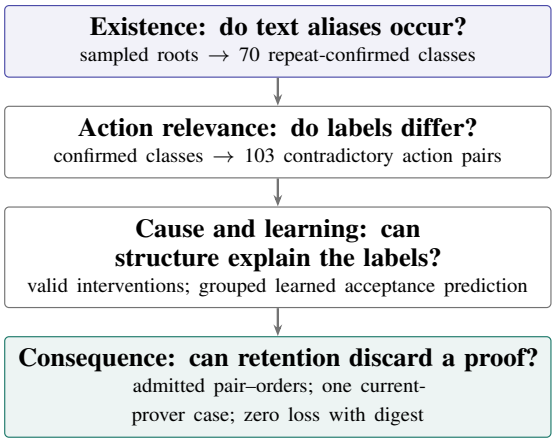


Figure 2: Conceptual evidence spine. Each arrow asks a separate question and uses the denominator printed beneath it. Population incidence, contradictory actions, structural learnability, and conditional search loss are estimated separately.

Dojo (Yang et al., 2023), we make three contributions:

1. **Impossibility.** We give repeat-verified counterexamples to action sufficiency: 68 natural alias classes contain opposite acceptance labels for the same tactic, yielding 103 contradictory textual inputs (§2–4). This conclusion is independent of the frequency with which a current policy encounters an alias.

2. **Empirical phenomenon.** Controlled interventions identify hidden elaborated structure as causal. A small acceptance model learns to use a compact structural channel, with theorem-grouped and mechanism-grouped evaluations separating interpolation from mechanism transfer (§5–7).

3. **Engineering consequence and repair.** A current step-prover and a 96-pair alias-conditioned stress benchmark demonstrate proof loss under unsafe retention. A structural identity key prevents every controlled loss at modest cost (§8).

Blind incidence, candidate exposure, and broad paired searches estimate the separate conditions under which the representational failure affects a search. Data, harnesses, and per-class outcomes are released.

2 Problem and scope

Let $T(s, a)$ denote the result of executing tactic a at Lean tactic state s, and Ψ the observable out-

come (acceptance with a successor, rejection, or proof closure). Rejection is what the interaction API reports as an error, including Lean's `made no progress`, the outcome most of the acceptance contradictions below take. A policy consumes $\varphi(s)$, the printed state; write $s_1 \sim_\varphi s_2$ when $\varphi(s_1) = \varphi(s_2)$.

The observation φ is *action-sufficient* if $s_1 \sim_\varphi s_2$ implies $\Psi(T(s_1, a)) = \Psi(T(s_2, a))$ for all a. This is a necessary one-step transition-consistency condition for abstractions such as bisimulation quotients and MDP homomorphisms (Givan et al., 2003; Li et al., 2006), which impose still stronger preservation requirements. We test only this one-step condition; its failure rules the stronger properties out. We distinguish acceptance, visible successor, and bounded-continuation equivalence throughout.

Acceptance ambiguity. For alias class C, some a has $r(s_i, a) \neq r(s_j, a)$, where $r \in \{0, 1\}$ is acceptance. The pair $(\varphi(s), a)$ then carries contradictory verified labels, and any acceptance predictor conditioned on it errs on at least $\min(n_0, n_1)/(n_0 + n_1)$ of members under the empirical member distribution. This lower bound applies to prediction of a specified action. The existence of a common successful tactic is a separate property.

Thus the corpus supplies contradictory supervision that the compiler certifies: Lean deterministically validates both labels on restored executable states, so each contradiction is a reproducible property of the data. A model given only the text can learn the average label distribution for an aliased input, while prediction specific to a member requires an additional observation channel.

Candidate exposure and a direct test on the model are reported in §7; the main text then evaluates search consequences.

Implementation scope. The observation claim is independent of a search implementation. The search-system claim is narrower: in the evaluated stack LeanDojo's `TacticState` compares and hashes on the rendering with the executable handle excluded (`compare=False`), and the best-first prover keys nodes on that value, so $\sim_\varphi$-equivalent nodes merge. LeanDojo-v2 uses Pantograph state handles and a history stack in its active prover, which avoids this particular merge. Its model prompt still uses the rendered state without the handle, so the action-sufficiency result continues to apply to policy conditioning. Appendix C gives the source and behavioral audits. A merge costs a proof when the discarded member's bounded continuation language contains a proof absent from the retained member's language.

3 Related work

Proof-state interfaces. LeanDojo (Yang et al., 2023) and LLMSTEP (Welleck and Saha, 2023) exemplify Lean interfaces that send a visible proof state to a language model. miniCTX (Hu et al., 2025) shows that file context beyond that state improves proving across problems. Our within-theorem members share the file, declaration, statement, and retrievable premises, so that context is identical. Earlier systems expose richer inputs: CoqGym's ASTactic encodes term trees (Yang and Deng, 2019), while TacTok combines proof-state semantics with partial proof-script history (First et al., 2020). A history-conditioned policy can distinguish the two arrivals studied here, although a merged node must still represent both members. LeanDojo, Pantograph (Aniva et al., 2025), and LeanTree (Kripner et al., 2025) define model or search interfaces without testing action sufficiency.

Structural and transition-grounded representations. Structural encodings include graph representations of higher-order logic (Paliwal et al., 2020) and Graph2Tac for Coq definitions (Blaauwbroek et al., 2024). Passport improves Coq proof synthesis with identifier categories, subwords, and paths (Sanchez-Stern et al., 2023). For Lean premise selection, Petrovčič et al. (2025) combine textual state embeddings with dependency graphs. These approaches enrich syntax, identifiers, or library context. Our paired members hold library context fixed and differ in elaborated structure local to the member. Samoylov and Vosoughi (2025) represent tactic semantics through induced state transformations, and 3D-Prover (Lamont et al., 2025) filters candidates by learned transition representations. Olejniczak et al. (2026) exploit proof-state symmetries. The present question is whether the visible state defines the transition being modeled. Contradictory deterministic transitions show that it need not. The same counterexample limits any tactic equivalence metric computed from text, such as LeanTutor's (Patel et al., 2025).

Partial observability and search abstraction. Goal-clone detection identifies alpha-equivalent Rocq goals to expose proof reuse opportuni-

	blind	coerc.	struct.
roots replayed	1,025	1,423	3,882
expandable	644	947	2,553
with reconvergence	112	234	667
rendering-identical classes	150	321	893
flagged divergent	11	50	36
repeat-confirmed	**3**	**47**	**21**
new after deduplication	3	47	20
acceptance-ambiguous	3	46	19
contradictory pairs	6	78	19

Table 2: Discovery by sampling regime. Only the blind column estimates incidence; targeted columns provide examples for mechanism analysis. One cross-regime duplicate leaves 70 unique classes.

ties (Ghanbari, 2025). Our operational criterion asks whether an equality used for policy input or node merging preserves action outcomes. Observation aliasing is the classical failure of a policy over observations to see its transition state (Whitehead and Ballard, 1991; Singh et al., 1994; Kaelbling et al., 1998). Tactic searches deduplicate states, the proof-search analogue of a transposition table (Zobrist, 1970); unsafe merging is the graph-history interaction problem (Kishimoto and Müller, 2004). Here elaboration creates the path dependence; game repetition rules are irrelevant.

4 Failure: search-reachable aliases

Method. From a root state we expand with a fixed tactic pool augmented by goal-derived rewrites and record, for every rendering, the set of *canonical* histories reaching it, where a step joins a history only if it changes the rendering (Appendix B). Classes reached by two prefix-free canonical histories are probed: one live state per history in a single session, the same battery for all members, and every flagged class re-verified on a fresh session with three repetitions per member. States are read at five levels of printer explicitness by metaprogram probes through the ordinary interaction API (Appendix B); derived digests separate the elaborated goal (F_{core}), the environment (F_{env}), and executable identity. Configuration: Mathlib `29dcec07`, Lean `4.10.0-rc1`, LeanDojo 2.0.2. The structural column combines two targeted sweeps. The second selected 2,400 roots and restored 2,234; expansion counts exclude failed restorations.

Table 2 separates incidence from discovery. In the blind sample, 112/644 expandable roots reconverge to an identical rendering, but only 3 classes on 3/1,025 sampled roots are confirmed operationally divergent: 0.29% (95% CI 0.06–0.85%). The targeted regimes contribute 67 more unique classes and support mechanism analysis only. In total, the 70 classes span 53 theorems, 67 roots, and 40 files in nine top-level Mathlib directories. A corpus-wide cross-file census is dominated by declaration context and is reported separately (Appendix A).

Human scripts rarely revisit identical renderings; the aliases studied here arise primarily through search reconvergence (Appendix A).

Acceptance is contradictory. In 68 of 70 classes some tactic is accepted on one member and rejected on another, stably across three repetitions on fresh sessions; the remaining two diverge only in successor states. This yields 103 contradictory observation–action pairs: identical model input, identical tactic text, opposite verified acceptance. Any acceptance predictor conditioned on $(\varphi(s), a)$ errs on at least $\min(n_0, n_1)/(n_0 + n_1)$ of members per pair. Over the 103 pairs the member-weighted aggregate is 0.40, a conditional inconsistency floor restricted to these pairs. The phenomenon spans several elaboration mechanisms, whose distributions are reported below.

5 Cause: hidden elaborated structure

Reading each class at five levels of printer explicitness localizes the hidden content (Appendix B). The consumed rendering φ_{state} and its depth-expanded variant fail to separate any class, and the environment digest F_{env} also fails wherever the probe completes. This excludes ambient declaration context as the cause. Disabling notation under `pp.all` separates 67 of 70, showing that the notation layer collapses most of the hidden content (Ullrich and de Moura, 2020); depth elision has no effect. The hidden content falls into six named categories plus two opaque classes: `ENNReal` coercion (56 classes), `Fin` eta-expansion (4), local-definition unfolding (4), instance paths (2), `Ne` versus `Not (Eq ...)` (1), and a proof term difference (1). In the dominant family the same token pair separates the members every time: a sixteen-byte difference in 48 of the 53 classes where the rung is measurable, and a larger one where the token recurs. The divergence follows from tactic implementation: `simp`'s progress check compares the resulting term syntactically, so one call can rewrite one member and report no progress

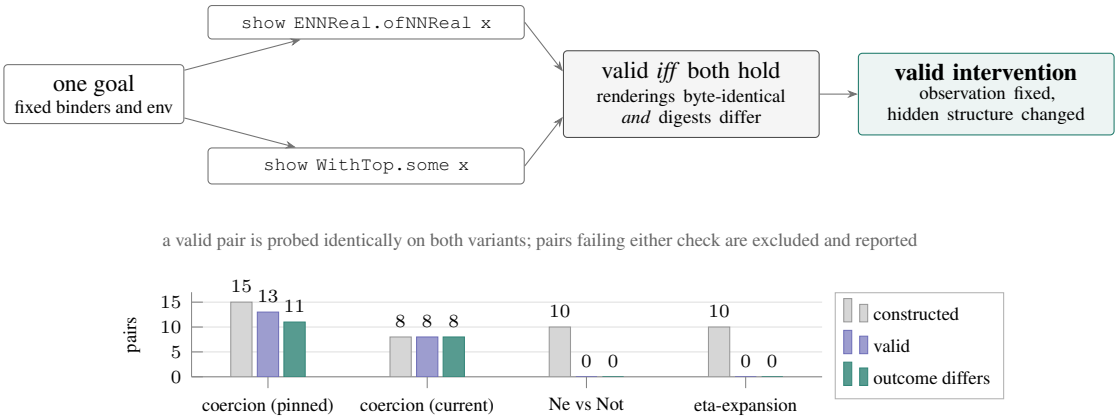


Figure 3: Rendering-preserving interventions. **Top:** the construction. A single `show` step replaces the goal with a definitionally equal term, so only the elaborated representation changes; a pair is used only if it passes both checks. **Bottom:** outcomes. The construction succeeds for the coercion family under the pinned stack and, with acceptance-sensitive probes, on a current Lean and Mathlib, where all 8 pairs are accepted on one variant and rejected on the other. The other two families yield zero valid pairs because the printer renders their two spellings differently.

on the other. Its direction depends on the simp set. A lemma stated for the abbreviation, as in the interventions below, rewrites only the member holding it. The default set rewrites the unfolded form: on the Table 1 witness, `simp` succeeds on the unfolded member with a successor that renders identically, and makes no progress on the member already holding the abbreviation.

Rendering-preserving interventions. The classes above are observational. To manipulate the hidden structure directly we construct pairs of standalone examples (Figure 3) identical in every respect except a single `show` step replacing the goal with a definitionally equal term carrying different hidden syntax: same environment, binders, goal statement, and probe text. `show` accepts any term definitionally equal to the goal, so it alters the elaborated representation without altering what is proved. Validity is verified per pair in both directions: the intervention must have taken effect, so the variants' structural digests must differ, and it must have remained hidden, so their pre-probe renderings must be byte-identical. Pairs failing either check are excluded and reported.

For the dominant coercion family 13 of 15 pairs pass both gates, and 11 of 13 change the probe outcome. The attempted `Ne` and eta interventions fail the byte-identity gate because their source spellings remain visible. The intervention therefore establishes a causal mechanism for the dominant family; the other observed families remain observational.

Visibility under continuation. In the original 59-class corpus, a shared continuation makes 18 classes visibly different, including 17 at `contrapose!`. A reduced battery that omits this operation yields near-total confinement. Escape is therefore a property of both the state and the tactic distribution (Appendix A).

Current stack. We repeat blind discovery on Lean `v4.33.0-rc1` and contemporaneous Mathlib. Sixty uniformly sampled compiled files yield 2,507 states. From 600 fixed-seed roots, 570 restore successfully and 98 rendering-alias classes are found. Fresh repeated probing confirms two classes at one root, with seven contradictory class–action pairs. The resulting root incidence is 1/570, or 0.18% (Clopper–Pearson 95% CI 0.004–0.97%). Both classes occur in `ModuleCat.disjoint_span_sum`; explicit printing separates terms produced by `rw` and `simp only`. In addition, all eight current coercion interventions pass both validity gates and reproduce opposite probe acceptance. Historical script replay reproduces one of eight older witnesses; the remaining seven fail after one lemma rename and six cases of API drift (Appendix C).

6 Learning from a structural channel

The observation aliases establish that the default rendering omits information relevant to the transition. We next train a deliberately small model to predict tactic acceptance from a compact structural channel local to the member. This test asks whether

a learned predictor can use information unavailable in the rendering. Candidate exposure and zero-shot use by a frozen prover are evaluated separately in §7.

Representation. For each restored member we extract a sorted multiset of application-head constant paths from a shallow explicit rendering of the local context and goal. This *structural sketch* restricts its features to structural constants. Tactics, pair identities, histories, theorem names, and outcome labels are excluded. The sketch compresses the explicit rendering before it is given to the evaluator.

Learned acceptance model. The extractor succeeds for all 178 members of the 70 confirmed classes. The acceptance evaluator selects one accepted and one rejected member for each repeat-confirmed class–action contradiction, giving 206 examples in 103 pairs from 51 theorems. We fit logistic regression over explicit sketch features and action–feature crosses (L_2=10^{-3}, 500 gradient steps). Each split constructs an explicit sparse vocabulary from its training examples alone. Coordinates first observed in the test split are omitted and receive zero weight. The theorem-grouped protocol keeps every example from one theorem in a single fold. It estimates transfer to unseen theorems while allowing mechanisms to recur across folds. The mechanism-grouped protocol defines groups from the hidden elaborated delta: coercion construction, `Fin` eta-expansion, instance path, local-definition unfolding, and the residual printer-blind classes. Each group is held out in turn. The mechanism macro weights the four audited mechanisms equally; the all-group macro also includes the residual group. The `Ne`-versus-`Not` and proof-term classes yield different successful successors without opposite acceptance labels, so they fall outside this binary evaluation. The paired metric assigns 1 when the accepted member receives the higher score, $1/2$ for a tie, and 0 for a reversal. Confidence intervals use the theorem as the resampling unit.

Within-mechanism generalization. Under theorem-grouped evaluation, the shallow sketch correctly orders 97 pairs, ties 5, and reverses 1, giving a paired rank of .966 (bootstrap over theorem clusters, 95% CI .927–.995). All 78 coercion pairs are correctly ordered. The default rendering remains constant within every pair and therefore scores .500. The structural digest separates all classes and serves only as an opaque identity key.

Representation	classes split	pair rank	accuracy
default rendering	0/70	.500	.500
structural sketch	67/70	**.966**	.850
structural digest	70/70	n/a	n/a

Table 3: Theorem-grouped learned acceptance prediction on 103 contradictory pairs. Pair rank measures whether the accepting member is scored above the rejecting member. The digest is excluded from model features.

Held-out group	classes	pairs	pair rank
`Fin` eta	4	10	1.000
instance path	2	7	.857
coercion	56	78	.500
local unfolding	4	4	.625
residual	2	4	.500
mechanism macro			.746
all-group macro			.696
pair weighted			.578

Table 4: Leave-one-group-out transfer with a fold-local explicit vocabulary. Each row trains on all other groups. The mechanism macro weights the four audited mechanisms equally; the all-group macro adds the residual group. The pair-weighted score retains the corpus distribution.

Mechanism transfer. The recurring coercion mechanism contributes 56 of 68 evaluated classes and 78 of 103 pairs. When coercion is held out, none of its 78 pairs contains a member-distinguishing coordinate observed in the training mechanisms; all 78 pairs therefore tie. The held-out `Fin` and instance-path groups score 1.000 and .857, and all 17 pairs contain a distinguishing coordinate observed during training. Local unfolding scores .625; three of its four pairs have such a coordinate. These results localize transfer to mechanisms that share informative structural coordinates with the training groups.

The explicit map removes collision and dimension choices. Across $L_2 \in \{10^{-4}, 10^{-3}, 10^{-2}\}$, theorem-grouped rank ranges from .956 to .966 and the pair-weighted mechanism holdout ranges from .558 to .578. The mechanism macro ranges from .621 to .746, chiefly because the four-pair local unfolding group ranges from .125 to .625. The theorem-grouped result supports interpolation across unseen theorems within observed mechanisms. The mechanism-held-out analysis establishes partial transfer for two small audited mecha-

Input	all	coerc.	`Fin`	inst.	unfold	macro
ordinary rendering	.500	.500	.500	.500	.500	.500
+ structural sketch	.539	.564	.400	.429	.625	.504

Table 5: Frozen-prover paired ranking on 103 confirmed tactics. Each entry is the fraction for which the accepting member scores above the rejecting member, with ties assigned one half. The macro weights the four audited groups of hidden deltas equally.

nisms and chance performance for the dominant unseen mechanism. Broader generalization requires training evidence that spans additional hidden structural deltas.

Across all 178 members, the median sketch is 934 bytes (maximum 2,503), compared with 4,212 bytes (maximum 282,108) for shallow `pp.all`. Shallow sketches distinguish 67 of 70 classes and the digest distinguishes all 70. The learned model therefore establishes compact identifiability of the hidden signal within observed mechanisms. The prototype currently obtains an explicit rendering before compression, leaving direct extraction as an engineering problem.

7 Direct evaluation on a frozen prover

The learned evaluator in §6 uses information in the sketch after training on the task. We next test whether a current prover can use that channel without adaptation, using the same paired acceptance task.

Protocol. We freeze BFS-Prover-V2-7B (Xin et al., 2025) at one immutable snapshot. For each of the 103 repeat-confirmed class–action contradictions, the model scores the same tactic on one accepting and one rejecting member. The ordinary arm uses the model-card prompt `{state}:::`. The structural arm places the sketch for that member in a Lean comment between the rendering and the separator. We average the conditional log probabilities of the tactic tokens and count whether the accepting member receives the higher score; ties receive one half. The checkpoint, prompt, and scoring rule remain fixed. No outcome label, theorem, class, or mechanism from the alias corpus enters experiment-side training or calibration. We report every group of hidden deltas separately and macro-average the four audited groups. Confidence intervals resample the 51 theorems.

Result. The structural input obtains .539 paired accuracy (bootstrap over theorem clusters, 95% CI .440–.644), with 53 correct rankings, 5 ties, and 45 reversals. Its macro average across the four audited mechanisms is .504. The ten pairs with equal augmented-prompt token counts score .550. The ordinary arm ties exactly on every pair because the rendering and tactic are byte-identical. Appending the sketch to this frozen prover provides no reliable acceptance-sensitive signal. Together with the trained model's .966 theorem-grouped score, this result decomposes structural-channel utility into information availability and model adaptation. The experiment conditions on a confirmed tactic; candidate generation and proof rate at the theorem level remain outside its estimand.

k	mean $\lvert A\rvert$	member cov.	ambiguous	headroom
1	1.0	.376	1	.000
2	2.0	.798	2	.000
4	4.0	.882	2	.000
8	7.5	.938	2	.000
16	11.9	.938	2	.000

Table 6: Candidate exposure over nested miniCTX beam prefixes. "Ambiguous" counts classes with an acceptance-divergent candidate. Headroom is the member-aware oracle coverage minus the best shared-action coverage. All 6,354 compiler calls agree within each state–candidate unit.

Candidate exposure. One deterministic miniCTX (Hu et al., 2025) beam-16 list is generated for each of the 70 class renderings, with a 16-token tactic cap. Nested prefixes give $k = 1, 2, 4, 8, 16$, and every candidate is executed three times on all 178 members. Table 6 shows that wider prefixes improve general coverage through $k = 8$, while acceptance-divergent exposure remains confined to two classes. The member-aware and best shared-action oracles have identical coverage at every width, giving zero member-specific headroom. Thus the structural channel identifies member differences that the present candidate sets do not convert into a distinct available proof choice. Fixed beam-8 results on the original corpus for miniCTX and two additional generators are reported in Appendix E.

8 Consequence: search and retention

Aliasing matters to search only when a policy reaches a class and continuation quality differs across its members. We evaluate conditional loss after an alias encounter, natural continuation loss, and broad policy incidence separately.

	budget			
Proposal generator	2	4	8	12
pairs proved by the refined arm (of 96)				
targeted	96	96	96	96
context bank	36	72	96	96
generic first	96	96	96	96
rendering loss, % of covered pair–orders				
targeted	48.4	48.4	48.4	48.4
context bank	50.0	50.0	50.0	48.4
generic first	31.3	31.3	31.3	31.3

Table 7: Exact results under uniform order on 96 admitted stress pairs. Loss is conditioned on the refined arm exposing a proof within budget. For the targeted generator, the pair-clustered 95% bootstrap interval is 46.4–50.0%; the eight-base-template interval is 46.9–49.5%.

Representative sensitivity. Every member in the original 59-class corpus is restored and searched independently under the same budget, model, and seed. With BFS-Prover-V2-7B (Xin et al., 2025), the first candidate list is byte-identical in all 59 classes; tactic execution alone then creates different trajectories. A proof is found in only two classes, and in only one do the members' outcomes differ. In `snormEssSup_add_le`, the `intro h; simp` member is proved at expansion 14; the `simp; intro h` member remains unproved across repetitions and at triple budget. A rendering-keyed merge can therefore discard the member proved under this continuation. An ancestor-rooted control proves through another intermediate under either retention order, localizing the observed loss to the aliased node (Appendix D).

Alias-conditioned stress benchmark. We cross eight current-stack, rendering-preserving intervention pairs with 12 proposition contexts. Admission requires Lean to verify byte-identical default renderings, distinct structural digests, and a fixed continuation that closes one member and fails on the other. All 96 pairs pass. After both branches reach the alias, rendering identity retains the first arrival; rendering plus digest retains both. We enumerate both arrival orders exactly and also run 200 seeded random orders per pair under three proposal generators and four budgets.

The targeted generator exposes the known continuation and `rfl`. The refined arm proves all 96 pairs under both orders. Rendering identity loses the proof for 93 of 192 pair–order combinations, giving the exact probability under uniform order $93/(2 \cdot 96) = 48.4\%$; the randomized estimate at budget 2 is 48.0%. Table 7 shows that context-bank and generic-first generators change the conditional rate. This variation identifies proposal exposure as part of the estimand. The benchmark establishes a conditional failure of bounded-search completeness under controlled alias encounters. Its shared coercion mechanism prevents a population-level interpretation (Appendix D).

Broad nulls. Paired searches with rendering-only versus structurally refined identity prove the same theorem in all 158 runs (36 proved in both; one-sided 95% upper bound 1.9% for discordance). Only 4 runs actually merge fingerprint-distinct states. Probing all 25 such merge pairs found no operational disagreement (upper bound 11.3%). Separately, replaying the known remainder of a Mathlib proof on 113 aliases loses none (upper bound 2.6%). These results constrain the broad rate under the tested policies and budgets. The 7B member search on the original corpus is outcome-informative in 2 of 59 classes and loses one local continuation. Policy exposure and candidate set regret are reported in §7 and Appendix E.

Identity refinement. Model observation and node identity can differ. A 64-bit hash of the goal closed over its context and normalized for metavariables separates all 70 classes and prevents every stress-benchmark loss. In a rotated 24-theorem cost test, verbose printing raises end-to-end time by 179%, versus 8% for the digest (Appendices B and C). The digest is a prototype: it is specific to one Lean version, can collide in principle, and provides no proof of operational equivalence. It serves as an opaque search identity, while the structural sketch in §6 is a feature for the model.

9 Implications for prover design

Separate interfaces. Pretty printing serves human communication and policy conditioning. Deduplication requires an identity with stronger transition-consistency properties. The results support separate representations for these functions: a stable textual observation for the policy and a structurally refined key for the search graph. Version tagging and collision checks remain necessary for a production identity scheme.

Structural conditioning. The paired-ranking result motivates a compact structural channel alongside the rendering. The sketch is local to the mem-

ber, generalizes across unseen theorems within observed mechanisms, and is substantially smaller than shallow `pp.all`. The trained acceptance model and the frozen prover intervention together locate the present bottleneck: the signal is learnable under training on the task, while an unadapted prover gives near-chance action scores. Direct extraction from Lean expressions could remove the prototype's explicit-printing stage. The digest remains reserved for opaque search identity.

Candidate generation. Member-specific scoring can affect proof search when the candidate set contains a corresponding choice. Increasing miniCTX beam width raises shared-action coverage but produces no member-specific oracle headroom through $k = 16$. Structurally conditioned generation or targeted proposal mechanisms are therefore the next experimental step. Candidate exposure and candidate set regret should accompany top-k accuracy when evaluating such systems.

10 Conclusion

Hidden elaborated structure produces contradictory deterministic labels for identically rendered state–tactic inputs, disproving action sufficiency for the standard textual observation. A compact structural channel makes the missing signal learnable within observed mechanisms, and controlled retention experiments show that rendering identity can discard an available proof while a structural key preserves it at modest cost. Human-readable rendering and machine-state identity serve distinct design objectives. Using one in place of the other fails transition safety; the demonstrated failure is removed by structurally refined identity.

Limitations

Representation and mechanism. The central claim concerns action sufficiency, for which one verified counterexample is decisive. The 70-class corpus characterizes the phenomenon under the reported search protocol and remains concentrated: 56 classes involve coercion construction, with four `Fin` eta, four local-unfolding, two instance-path, two residual, one `Ne`-versus-`Not`, and one proof term class. Causal identification applies to the dominant family. Theorem-grouped prediction measures interpolation within observed mechanisms. Mechanism-held-out transfer depends on shared training coordinates, and the dominant unseen coercion group ties at chance.

Models and search. The test on the frozen prover evaluates one prompt and scores confirmed tactics, and it is underpowered: its interval spans the null, so it bounds a small zero-shot effect without excluding one. Its scope is this checkpoint and this prompt, which deliver no usable signal specific to a member. Whether another frozen model would remains untested. The checkpoint may have encountered Mathlib during training. Candidate-set analyses measure exposure within fixed decoded lists, while proof-rate effects also depend on generation, search policy, and budget. The stress benchmark estimates conditional completeness loss after an admitted alias encounter with an available member-separating continuation. Its 96 pairs derive from one mechanism crossed with logical contexts. Blind sampling and broad paired searches estimate encounter frequency under the evaluated distributions; the single natural loss concerns the aliased node's bounded continuation.

Scope beyond this stack. Every measurement here is confined to Lean 4, Mathlib, and the LeanDojo interface. The two preconditions are general, namely a printer that suppresses elaborated distinctions and an interface that exposes only its output, and other assistants and their neural interfaces satisfy them in outline. We have not measured any of them, so whether comparable aliases arise in Rocq, Isabelle, or non-LeanDojo Lean interfaces remains untested.

References

Leni Aniva, Chuyue Sun, Brando Miranda, Clark Barrett, and Sanmi Koyejo. 2025. Pantograph: A machine-to-machine interaction interface for advanced theorem proving, high level reasoning, and data extraction in Lean 4. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*.

Kshitij Bansal, Sarah Loos, Markus Rabe, Christian Szegedy, and Stewart Wilcox. 2019. HOList: An environment for machine learning of higher order logic theorem proving. In *International Conference on Machine Learning (ICML)*, pages 454–463.

Lasse Blaauwbroek, Miroslav Olšák, Jason Rute, Fidel Ivan Schaposnik Massolo, Jelle Piepenbrock, and Vasily Pestun. 2024. Graph2Tac: Online representation learning of formal math concepts. In *International Conference on Machine Learning (ICML)*.

Emily First, Yuriy Brun, and Arjun Guha. 2020. TacTok: Semantics-aware proof synthesis. *Proceedings of the ACM on Programming Languages*, 4(OOPSLA):231:1–231:31.

Ali Ghanbari. 2025. Automatic goal clone detection in Rocq. In *39th European Conference on Object-Oriented Programming (ECOOP)*, volume 333 of *Leibniz International Proceedings in Informatics*, pages 12:1–12:19.

Robert Givan, Thomas Dean, and Matthew Greig. 2003. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1–2):163–223.

Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward W. Ayers, and Stanislas Polu. 2022. Proof artifact co-training for theorem proving with language models. In *International Conference on Learning Representations (ICLR)*.

Jiewen Hu, Thomas Zhu, and Sean Welleck. 2025. miniCTX: Neural theorem proving with (long-)contexts. In *International Conference on Learning Representations (ICLR)*.

Albert Q. Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. 2023. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *International Conference on Learning Representations (ICLR)*.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134.

Akihiro Kishimoto and Martin Müller. 2004. A general solution to the graph history interaction problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 644–649.

Matěj Kripner, Michal Šustr, and Milan Straka. 2025. LeanTree: Accelerating white-box proof search with factorized states in Lean 4. *arXiv preprint arXiv:2507.14722*.

Sean Lamont, Christian Walder, Amir Dezfouli, Paul Montague, and Michael Norrish. 2025. 3D-Prover: Diversity driven theorem proving with determinantal point processes. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Guillaume Lample, Marie-Anne Lachaux, Thibaut Lavril, Xavier Martinet, Amaury Hayat, Gabriel Ebner, Aurélien Rodriguez, and Timothée Lacroix. 2022. HyperTree proof search for neural theorem proving. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Lihong Li, Thomas J. Walsh, and Michael L. Littman. 2006. Towards a unified theory of state abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*.

Jannis Limperg and Asta Halkjær From. 2023. Aesop: White-box best-first proof search for Lean. In *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*.

Leonardo de Moura and Sebastian Ullrich. 2021. The Lean 4 theorem prover and programming language. In *Automated Deduction (CADE)*.

Krzysztof Olejniczak, Radoslav Dimitrov, Xingyue Huang, Bernardo Cuenca Grau, Jinwoo Kim, and İsmail İlkan Ceylan. 2026. What are the right symmetries for formal theorem proving? *arXiv preprint arXiv:2605.22257*.

Aditya Paliwal, Sarah Loos, Markus Rabe, Kshitij Bansal, and Christian Szegedy. 2020. Graph representations for higher-order logic and theorem proving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2967–2974.

Manooshree Patel, Rayna Bhattacharyya, Thomas Lu, Arnav Mehta, Niels Voss, Narges Norouzi, and Gireeja Ranade. 2025. LeanTutor: Towards a verified AI mathematical proof tutor. *arXiv preprint arXiv:2506.08321*.

Job Petrovčič, David E. Narváez, and Ljupčo Todorovski. 2025. Combining textual and structural information for premise selection in Lean. In *NeurIPS Workshop on Mathematical Reasoning and AI (MATH-AI)*.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*.

Elisaveta Samoylov and Soroush Vosoughi. 2025. Modeling tactics as operators: Effect-grounded representations for Lean theorem proving. In *Proceedings of the 3rd Workshop on Mathematical Natural Language Processing (MathNLP)*, pages 168–175.

Alex Sanchez-Stern, Emily First, Timothy Zhou, Zhanna Kaufman, Yuriy Brun, and Talia Ringer. 2023. Passport: Improving automated formal verification using identifiers. *ACM Transactions on Programming Languages and Systems*, 45(2):12:1–12:30.

Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. 1994. Learning without state-estimation in partially observable Markovian decision processes. In *International Conference on Machine Learning (ICML)*, pages 284–292.

The mathlib Community. 2020. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*.

George Tsoukalas, Jasper Lee, John Jennings, Sangeetha Xin, Michelle Ding, Michael Honerkamp, Kyunghoon Kim, and Swarat Chaudhuri. 2024. PutnamBench: Evaluating neural theorem-provers on the Putnam mathematical competition. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*.

Sebastian Ullrich and Leonardo de Moura. 2020. Beyond notations: Hygienic macro expansion for theorem proving languages. In *Automated Reasoning (IJCAR)*, pages 167–182.

Sean Welleck and Rahul Saha. 2023. LLMSTEP: LLM proofstep suggestions in Lean. In *NeurIPS Workshop on Mathematical Reasoning and AI (MATH-AI)*.

Steven D. Whitehead and Dana H. Ballard. 1991. Learning to perceive and act by trial and error. *Machine Learning*, 7(1):45–83.

Ran Xin, Zeyu Zheng, Yanchen Nie, Kun Yuan, and Xia Xiao. 2025. Scaling up multi-turn off-policy RL and multi-agent tree search for LLM step-provers. *arXiv preprint arXiv:2509.06493*.

Kaiyu Yang and Jia Deng. 2019. Learning to prove theorems via interacting with proof assistants. In *International Conference on Machine Learning (ICML)*, pages 6984–6994.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. miniF2F: A cross-system benchmark for formal Olympiad-level mathematics. In *International Conference on Learning Representations (ICLR)*.

Albert L. Zobrist. 1970. A new hashing method with application for game playing. Technical Report 88, Computer Sciences Department, University of Wisconsin. Reprinted in *ICCA Journal*, 13(2):69–73, 1990.

A Corpus context and natural occurrence

Cross-file census. Across 168,074 auditable Mathlib states, 271 observation-collision classes diverge operationally, but approximately 99% differ only in declaration context: the members occupy different files or positions, so their environments differ. State-weighted prevalence on elaborated-goal identity is 0.03%. An intervention control adding valid declarations while holding the elaborated goal and tactic fixed produced one success-to-failure transition in 497 trials. Cross-file classes cannot be merged by a transposition table, since two theorems never co-occur in one search tree, which is why the main text measures within-search aliasing.

Human proof trajectories. Across 60,665 traced Mathlib proofs and 168,074 states, only 10 proofs revisit a byte-identical rendering, yielding 12 position pairs. In all 12 the author applies a different tactic at the two positions, leaving zero natural same-tactic pairs for probing. Linear proof scripts thus rarely create the repeated observation that search produces through reconvergence.

Visible escape. Escape depth is the first step at which member renderings or terminal outcomes differ under a shared continuation, generated by the policy from the shared observation and applied to every member in lockstep to depth 8, stopping at the first divergence, terminal disagreement, or revisited rendering. Of 59 classes, 19 escape and 18 pass repeat controls. Successor diffs are identical in form: one member's goal becomes $\forall b,\ h < b \rightarrow \ldots$ while the other's becomes $\neg\exists b,\ \cdots \wedge b \leq h$.

B Measurement apparatus and protocol

Fingerprint ladder. One state is read at five levels of printer explicitness by `run_tac` probes: the default rendering φ_{state}; φ_{shallow} (`pp.all`, notation disabled); φ_{deep} (`pp.deepTerms`); φ_{proofs}; and φ_{full}. Two implementation properties matter for reproduction: `pp.all` leaves `pp.deepTerms` and proof-term suppression separately configurable, and module identity must be excluded from executable digests or every cross-file pair separates trivially.

Experimental specification. Policies: `leandojo-lean4-tacgen-byt5-small` and `ntp-mathlib-st-deepseek-coder-1.3b`, state-only input, inputs truncated at 2,300 tokens, deterministic beam search (8 beams), k=8; representative sensitivity additionally uses `BFS-Prover-V2-7B` (prompt `{state}:::`, deterministic beam-8, 24 expansions, 300 s per member, verification at 72 expansions and 900 s). Expansion pool: `simp`, `constructor`, `intro h`, `omega`, plus per-hypothesis `rw [h]` and `simp only [h]` ($\leq$16 per state), depth 2. Probe battery: `rfl`, `assumption`, `omega`, `simp`, `norm_num`, `constructor`, `exact?`, `trivial`, `decide`, `simp_all`, `tauto`, `aesop`, plus goal-derived rewrites; the merge-pair battery adds `contrapose!`, `push_neg`, `norm_cast`. Per-probe watchdogs 30–120 s; session timeouts 300–900 s; three repetitions per member; fixed seeds; shards deduplicated by (file, theorem, tactic index) and confirmed classes by (theorem, rendering hash).

For the direct test on the model, the frozen BFS-Prover-V2 checkpoint scores each confirmed tactic by mean conditional token log probability. The structural prompt appends that member's shallow sketch in a Lean comment before the model-card separator. The ordinary and structural arms use identical tactic tokens. The paired metric assigns

one, one half, or zero according to whether the accepting member scores above, ties, or scores below the rejecting member. The bootstrap over theorem clusters uses 10,000 resamples.

The width sweep uses miniCTX snapshot `7c00b1175a82e`, one deterministic 16-beam decode per class, and a 16-token cap. Prefixes of the single ranked list define each k, ensuring nested candidate sets. All candidates are executed three times per member; a failure includes parser, elaboration, and no-progress errors.

Controls against false positives. Three controls address hazards that produced false positives during development. After any watchdog interrupt the session is discarded, because channel desynchronization can return an earlier probe's answer. Every divergence must be self-consistent over three repetitions per member in a fresh session; this removed 11 of 13 candidates supported only by nondeterministic `aesop` (Limperg and From, 2023). Escape is reported relative to the tactic battery, and canonicalization based on the rendering was compared with a separating fingerprint on 369 paired roots. The artifact retains exclusions, repeat tables, and resource failure verdicts.

Structural digest. One 322-byte `run_tac` probe: `mkForallFVars (lctx.getFVars) type` closes the goal over its local context, converting free variables to de Bruijn indices; `instantiateMVars` resolves assignments; `abstractMVars` canonically renumbers remaining expression and universe metavariables; the resulting term's 64-bit `Expr.hash` is returned. Validated with three repetitions per member on all 59 classes in the original corpus (59/59 separated, all stable) and as a search key on 39 theorems. Table 3 extends the separation audit to the full 70-class corpus, where the digest separates 70/70 classes. The hash approximates normalized structural equality: collisions are possible in principle and the encoding is Lean-version specific.

C Node identity and persistence across versions

Version persistence. The blind audit on the current stack samples 60 of 8,302 compiled files uniformly with a fixed seed and traces all 60; they contain 989 theorems and 2,507 tactic states. Of the 600 sampled roots, 570 complete restoration and expansion, while 30 resource or restoration failures are recorded and excluded from the incidence denominator. The two confirmed classes have two- and four-member histories differing through `rw [huv]` and `simp only [huv]`; shallow `pp.all` and the structural digest distinguish both. Seven probes give stable opposite acceptance labels and no probe is nondeterministic.

Eight representative classes were replayed in place on Mathlib master `3b3cdbb` under Lean `v4.33.0-rc1`, splicing each proof back into its own file so namespace and variable context are preserved. One reproduces fully, with both histories elaborating, all four members' renderings byte-identical, and the probe's successors still diverging. Of the remaining seven, one is a pure rename: the Table 1 witness persists as `lintegral_ofReal_le_lintegral_enorm` with its prefix referencing a renamed lemma, and six are blocked by tactic-level API drift. Every class that remained constructible retained the divergence.

Node identity across package versions. In the evaluated stack (LeanDojo 2.0.2) `TacticState` is a frozen dataclass whose printed field participates in equality and hashing while the executable handle is excluded with `compare=False`, and the shipped best-first prover indexes nodes by that value. Directly constructed states with one rendering and different handles compare equal, hash equal, and collide in a dictionary.

We separately audit the published LeanDojo-v2 1.0.9 wheel (SHA-256 `635e8f99...`) and its active Pantograph backend at commit `f8aee320`. The active `BaseProver` keeps a history stack and contains no state-indexed transposition dictionary. Pantograph `GoalState` includes `state_id`; two constructed states with equal rendered goals and different identifiers render equally and compare unequally. They are also unhashable because their goal fields are lists. Thus the active v2 prover preserves these distinct live handles and avoids the evaluated rendering-keyed merge.

The v2 wheel also contains a legacy search tree module whose `InternalNode` compares a `TacticState`, but the wheel omits the interaction state implementation that module imports, so we base the conclusion on the active Pantograph path, which supplies the model with `str(GoalState)`. Because that string omits `state_id`, equal renderings can remain observation aliases for the policy. The search loss claim ap-

Identity	nodes	peak queue	time	probe share
rendering	230	135	1.00×	0%
shallow `pp.all`	233	138	2.79×	64%
structural digest	232	136	1.08×	15%

Table 8: Node-identity cost over 24 rotated-order theorem searches. Time is total wall clock relative to the rendering baseline. All arms prove the same 4 theorems; node and queue differences occur on 2 of 24.

plies to the evaluated LeanDojo prover; the action sufficiency claim applies to both textual interfaces.

Cost of a refined identity key. The cost test runs twenty-four theorems at 12 expansions, rotates arm order with the theorem index, and adds a baseline arm that omits the probe. The baseline is needed because the paired-search harness computes φ_{shallow} in every arm for instrumentation, so its recorded wall clock cannot isolate the key's cost. Node and queue counts appear in Table 8; each refined key differs from the baseline on 2 of 24 theorems. Expansion counts are nearly identical (150, 150, 151) because the budget caps them, so the counts are uninformative about node overhead. The printing probe costs 0.47 s per node at the median and consumes 64% of its arm's search time (paired median 2.55×, maximum 12.1×); the digest costs 0.090 s and consumes 15% (paired median 1.27×). Generation and tactic execution are unchanged across arms and all three arms prove the same 4 theorems.

D Retention protocol and reproducibility

Stress construction and admission. Eight acceptance-divergent interventions on Mathlib `3b3cdbb`/Lean `v4.33.0-rc1` are crossed with 12 proposition contexts, including conjunction, disjunction, equality with `True`, double negation, and duplicated goals. Each context has a logical introduction sequence that exposes the base pair's acceptance-sensitive probe as the proof-producing step. Lean compilation then enforces three admission conditions: byte-identical default renderings, distinct structural digests, and closure of the abbreviated member with rejection on the unfolded member. All 96 candidates pass. The harness compiles 192 capture declarations and a matrix of 17,472 state–action labels. Capture declarations log the state before an explicit `sorry`; no state–action closure label relies on `sorry`.

Retention and proposal protocol. The comparison begins after both branches reach the admitted alias. Rendering identity retains the first arrival, while rendering plus digest retains both. The targeted generator emits the pair's known continuation followed by `rfl`. The context-bank generator emits the eight continuations lifted through the same proposition context, followed by `rfl`. The generic-first generator prefixes `rfl`, `simp`, and `norm_num` to that bank. Budgets are 2, 4, 8, and 12 actions. Both orders are enumerated, and 200 uniformly randomized orders per pair and configuration use seed `731991`. The primary interval resamples pairs; a second interval clusters the 12 contexts by their eight base templates. Exact order enumeration eliminates Monte Carlo error. The bootstrap intervals measure sensitivity to the constructed pair set.

Supplementary controls. The earlier eight-pair protocol gives five order-dependent losses with the pair-specific grammar and the same five with one fixed union grammar over 194 compilations. For the natural case, injecting both histories from their common root and continuing with BFS-Prover-V2 proves every retention arm through another intermediate within 2–4 expansions. The observed natural loss is therefore confined to the aliased node's continuation.

Step-prover configuration. Checkpoint `ByteDance-Seed/BFS-Prover-V2-7B`, snapshot `533d59fc6b5e`, with its own tokenizer; prompt is the state's rendering followed by `:::` per the model card; float16 on Apple MPS. Decoding is deterministic beam search, 8 beams, 8 return sequences, at most 64 new tokens, early stopping; the continuation is decoded without special tokens, text after any re-emitted `:::` is kept, and candidates are deduplicated preserving the best sequence score. Tactic execution carries a 60 s watchdog; a watchdog interrupt abandons the class as channel desynchronization. The search queue orders by negated cumulative sequence score with insertion order breaking ties. Per-class first candidate lists, per-search reached-state hashes, and proof traces are in the released run files, together with the three repetition runs and the triple-budget run for the sensitive class.

E Candidate exposure across policies

Fixed candidate sets. For each of the 146 restored members in the original 59-class corpus, we execute the BFS-Prover-V2 top-8 list three

Policy	$\lvert C\rvert$	covered	ambiguous	rank 1	headroom > 0
ReProver	59	55	2	0	0
miniCTX	55	41	2	1	0
BFS-7B	59	56	1	0	0

Table 9: Candidate exposure. "Covered" counts classes with an accepted candidate for every member; "ambiguous" counts classes containing an acceptance-divergent candidate.

times, yielding 3,504 compiler calls and 1,168 state–candidate labels. The lists are identical across members of every alias class, and all repetitions agree.

Candidate set headroom. For candidate set A and class C, the member-aware oracle averages $\max_{a \in A} r(s_i, a)$ over members. The shared-action oracle maximizes the mean acceptance of one action across members. Their difference measures member-specific headroom within A and has no monotonic relationship with the full tactic language.

For BFS-Prover-V2, the member-aware and shared-action oracles both cover 134/146 members. The corresponding equality holds in every fully covered class for all three generators. These fixed candidate sets provide almost no opportunity for a member-aware score to alter proof availability.

F Artifact

The supplement is available at `https://github.com/edisonly1/mathnlp1`. It contains the harnesses, per-class outcomes, intervention files, policy candidates, search traces, pinned versions, and seeds. One command reproduces each table from the released records, with live modes for the motivating witness and the current intervention.

Scoring-Location Sensitivity in Process Reward Models

Jai Kumar Sharma
Virginia Tech
jaisharma@vt.edu

Abstract

Process reward models (PRMs) assign scores to reasoning steps, but an evaluator must also choose where to score. We ask whether that choice changes the predicted earliest error while the reasoning text stays unchanged. On 800 selected ProcessBench questions we compare a punctuation policy and a near-midpoint policy that insert equal numbers of within-step scoring locations, preserve every reasoning byte, and map decisions back to the original steps. At threshold 0.5 the two policies disagree on the predicted earliest error for 10.4%, 14.5%, and 14.4% of questions for Qwen2.5-Math-PRM, Math-Shepherd, and Skywork, and for the marker interfaces the disagreement exceeds a separator-only comparison on the same questions. The direct comparison was identified post hoc; a disjoint, prospectively frozen 400-question panel again shows 10.0–14.3% disagreement. Two mechanisms are at work: marker-based policies can change scores at the original step endpoints, whereas Skywork keeps every token score fixed and changes the predicted earliest error only through which positions are inspected. These are controlled sensitivity measurements, not prevalence estimates for naturally occurring segmenters. PRM evaluations should report marker rendering, separators, readout positions, the mapping to original steps, and the decision threshold.

1 Introduction

An earliest-error prediction is not determined by a PRM checkpoint and reasoning trace alone. The evaluator must also decide where to insert scoring markers or which token positions to inspect; we call this rule the *scoring-location policy*. Would the same model flag the same original step if only that policy changed while the reasoning text stayed fixed?

The question matters when PRM evaluations are compared. ProcessBench responses were flattened and resegmented before annotation (Zheng et al., 2025), marker conventions differ across scoring interfaces (Qwen Team, 2025; Zhang et al., 2025; Wang et al., 2024), and a recent survey lists standardized step definitions and evaluation protocols among the open problems (Zheng et al., 2026). We therefore compare two controlled policies on the same responses: one adds scoring locations at punctuation boundaries, the other places the same number of locations near the token midpoints of the same eligible steps. Both preserve the reasoning bytes, and every predicted error is mapped back to the original ProcessBench step. Figure 1 shows a response on which the two policies flag different original steps.

On the original 800-question panel the policies disagree on 10.4%, 14.5%, and 14.4% of questions for Qwen2.5-Math-PRM, Math-Shepherd, and Skywork, and for the marker interfaces the disagreement exceeds a separator-only comparison on the same questions. This direct comparison was identified post hoc on the original panel; we then froze it before scoring a disjoint 400-question panel, which also showed disagreement. The preregistered primary asked a different question, whether one policy differs from the reference more often than the other, and was null; §5.2 explains why that result coexists with direct policy disagreement.

We also ask how the decisions change. With marker-based interfaces, changing scoring locations can alter scores at the original endpoints. With Skywork, the token scores stay fixed, yet inspecting different positions still changes the predicted earliest error. The study measures sensitivity under controlled policies on selected questions, not its prevalence under naturally occurring segmenters; its practical implication is that a PRM evaluation should specify the scoring-location policy as well as the checkpoint.

Our contributions are (1) a controlled audit that preserves the reasoning text, matches the number of

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 254–267
October 29, 2026 ©2026 Association for Computational Linguistics

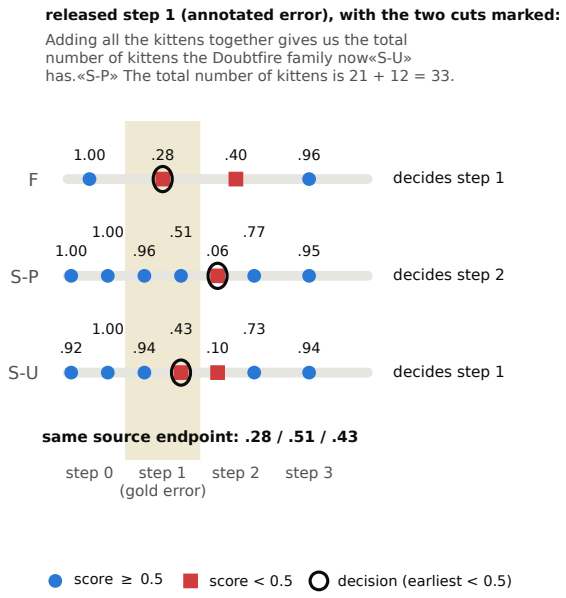


Figure 1: The same reasoning response yields different earliest-error predictions (ProcessBench row `gsm8k-118`, Qwen2.5-Math-PRM, fp32 canonical rescore). Top: released step 1 (the annotated error) with the punctuation cut (S-P) and the near-midpoint cut (S-U) marked at their recorded offsets. Bottom: block scores under F, S-P, and S-U; circles are scores at or above τ=0.5, squares below, and the ring marks the earliest below-threshold block (the decision). At the highlighted original step-1 endpoint Qwen assigns 0.28 under F, 0.51 under S-P, and 0.43 under S-U, so F and S-U flag step 1 while S-P flags step 2: the original endpoint score changed although the reasoning text did not.

scoring locations, and maps predictions back to the original steps; (2) measurements of direct policy disagreement with separator and local-position controls and a prospectively frozen disjoint panel; and (3) a diagnostic that separates changes to existing scores from changes caused by reading different positions of a fixed score sequence, with a reporting recommendation.

2 Related Work

ProcessBench, PRMBench, and Hard2Verify establish step-level verification benchmarks (Zheng et al., 2025; Song et al., 2025; Pandit et al., 2026), building on PRM800K's process supervision (Lightman et al., 2024).

Closest to us, EST-PRM stress-tests frozen PRMs with label-preserving structural transformations (Shihab et al., 2026), BiPRM perturbs ProcessBench step structure and reports F1 degradation in the perturbed representation (Zhang et al., 2026), and Beyond the First Error studies PRMs for reflective chains under differing step-segmentation conventions (Yang et al., 2025). Prior PRM robustness studies examine distribution shift, formatting, and adversarial perturbations, with findings that depend on the model and the perturbation type (Dontsov et al., 2026; Tiwari et al., 2026); more broadly, FormInv studies the semantic invariance of mathematical reasoning benchmarks under paraphrase (Thomas and Thomas, 2026). These studies do not hold the reasoning bytes fixed with exact source pullback, and they evaluate perturbations by aggregate reward, PRM@N, F1, semantic-consistency, or ranking metrics rather than by source-aligned paired earliest-error disagreement.

Training-time work varies the supervision unit (Liu et al., 2025; Hu et al., 2025; Cao et al., 2025; Wang et al., 2026). Token-level value models remove explicit markers but still need a declared rule for turning token scores into a discrete earliest-error decision (Chen et al., 2025; Lee et al., 2025; He et al., 2024), and uPRM interleaves trainable marker tokens (Gadetsky et al., 2026). Generative or reasoning-style verifiers (She et al., 2025; Zhao et al., 2026; Khalifa et al., 2026; Zhong et al., 2026) and policy-aware discriminative training (Agrawal et al., 2026) change how correctness signals are produced; our measurements cover only the three scalar interfaces below. Our audit differs from the closest prior work in holding the reasoning bytes fixed, comparing source-aligned earliest-error decisions directly between two count-matched scoring policies, adding a locally matched control, and diagnosing whether existing scores or only the inspected positions changed.

3 Audit Framework

3.1 Views, byte identity, provenance

A *released step* is one of the ProcessBench steps, a *scoring location* is a position at which a PRM score is obtained, and a *view* is one placement of scoring locations over a response. The reported comparisons use four views per response. The reference view, **F**, scores the released step boundaries. The separator control, **R0**, uses those same boundaries under the model-card separator convention. The **punctuation view**, S-P, adds one scoring location at an eligible clause boundary of every jointly admissible step, and the **near-midpoint view**, S-U, adds one location to the same steps at the admissible non-punctuation whitespace boundary nearest the token midpoint. F, S-P, and S-U preserve the

original reasoning bytes once the scoring markers are removed; R0 is the explicit exception, since it changes the separator rendering. All cuts are chosen once, without labels or rewards, and reused verbatim by every interface, so S-P and S-U have identical marker counts and split-step sets. For the marker interfaces the rendered model input changes when markers are added or moved; for Skywork one unchanged input is read at different positions, and F and R0 coincide (no separator intervention exists). Every visible block records its source-coordinate span and released-step interval (coordinate conventions, admissibility floors, fail-closed invariants, and the no-truncation audit: Appendix A; literal rules: Table 5).

Roles of the views. The split pair is a controlled stress test built for identification, not a sample from deployment policies; R0 supplies a realistic separator-only anchor, and naturalistic policy prevalence is outside this study (§9).

3.2 Decisions and conservative compatibility

For each view we predict the earliest visible block whose score is below 0.5; when no block is below the threshold the output is BOT, meaning "no error detected", not that the solution is correct. The flagged block is then mapped back to its original ProcessBench step. In the main F/S-P/S-U comparisons every flagged block maps to one original step, so two policies disagree exactly when they flag different original steps or when only one detects an error; the general rule below also handles multi-step blocks. Formally, a non-BOT decision maps through provenance to its inclusive released-step interval. Two decisions are *definitely incompatible* exactly when BOT status differs, or both are non-BOT and the intervals are disjoint; overlap counts as compatible. Formally, $I_x(V, W)$ indicates that event for row x, and the fixed-panel turnover is $\hat{d}(V, W) = n^{-1} \sum_i I_{x_i}(V, W)$; we reserve Pr for explicitly declared resampling targets.

3.3 Three scoring interfaces

Two interface families realize a scoring location differently. For Qwen and Math-Shepherd a scoring location requires a marker in the model input, so adding or moving locations changes the rendered input; for Skywork a scoring location selects a position in one already-computed token-score sequence, so the input never changes. **Marker token (Qwen).** Qwen2.5-Math-PRM-7B (Qwen Team, 2025) inserts its dedicated `<extra_0>` marker token after each block and reads the positive-class probability there. **Tag string (Shepherd).** Math-Shepherd (Wang et al., 2024) appends its tag string (space-prefixed, per the model card) and reads $P(+)/(P(+)+P(-))$ at each tag. In both, adding a scoring location changes the model input and later token history: the intervention is *adding scoring locations under a marker-input interface*, not count in isolation. **Readout-only (Skywork).** Skywork-o1-Open-PRM-1.5B (He et al., 2024) exposes a per-token value head, so all views share one byte-identical input and one forward pass, and a policy is an ordered readout set over that fixed field; readouts are mapped at token granularity, so this arm measures token-grid sensitivity. We use Skywork as a fixed-field mechanism control, not as a competitive verifier (rendering conventions and readout mapping: Appendix A; literal separators, tag strings, prompt serialization, checkpoint revisions, and cut rules: Table 5).

3.4 The fixed-field invariant

Suppose every original scoring location is retained and its score is unchanged. Inspecting additional locations may reveal an earlier low score, but it cannot remove a low score that was already present; so if the earliest detected error moves later or disappears, the score at the old decision endpoint must have changed. This directional statement needs the original endpoint set to be contained in the new one; it does not order S-P against S-U, whose added positions are not nested. Formally, with a fixed score field, adding readout positions can only preserve an existing earliest crossing or move it earlier; a move to a *later* endpoint or to BOT therefore certifies that a score at an existing endpoint changed (Lemma 1 and the upward-crossing certificate, Corollary 1, with proofs: Appendix A). Figure 1 is such a certificate case. This invariant is what lets the audit separate input-side field change from added-readout effects.

4 Study Design and Epistemic Status

Two panels. From the four ProcessBench subsets (GSM8K, MATH, OlympiadBench, Omni-MATH) we select 800 unique question keys, 200 per subset, that pass frozen eligibility gates (4–12 nonempty steps, bounded rendering length, joint split admissibility), decontaminated from a 20-response discovery pilot; selection is by pre-committed salted

	S-P vs. S-U		F vs. R0
Interface	Original (800)	Disjoint (400)	Original (800)
Qwen	83 (10.4%)	44 (11.0%)	40 (5.0%)
Shepherd	116 (14.5%)	57 (14.3%)	51 (6.4%)
Skywork	115 (14.4%)	40 (10.0%)	n/a

Table 1: Direct earliest-error disagreement between the punctuation and near-midpoint policies (S-P vs. S-U) at τ=0.5 (fp32) on the original and disjoint panels, and the separator-only comparison F vs. R0 on the original panel; headers give denominators. Qwen and Shepherd are marker interfaces; Skywork is readout-only and realizes F and R0 as one readout set on one input, so no separator intervention exists (n/a).

hash, one row per key (funnel and selection diagnostics: Appendix D). A disjoint panel of 400 keys (100 per subset) was frozen before any of its rows were scored. Macro F1 of the reference view lands within about four F1 points of published figures for both marker interfaces, a coarse implementation sanity check (Appendix B).

Analysis history. The central descriptive quantity is direct policy disagreement, $\hat{d}(S\text{-}P, S\text{-}U)$, compared with the separator-only rate $\hat{d}(F, R0)$ on the same questions. The preregistered primary was a different quantity, the difference of marginal reference-relative displacement rates $\hat{d}(S\text{-}P, F) - \hat{d}(S\text{-}U, F)$. The direct comparison was identified post hoc on the original panel and then frozen before the disjoint panel was scored; "frozen" means specified before that scoring run, not before any original-panel result was known. Exact finite-panel counts come first, intervals are descriptive sensitivity analyses, and Appendix D summarizes the amendment, deviation, and status record.

Numerical precision. All headline descriptive rates use a canonical rescore in 32-bit floating point (fp32); the preregistered primary stays on its locked bfloat16 (bf16) basis, and bf16 is otherwise retained as a sensitivity (Appendix C).

5 Scoring Policies Disagree on 10–15% of Earliest-Error Decisions

5.1 Direct policy disagreement

Original panel. Under the conservative compatibility rule, the two equal-count policies produce incompatible earliest-error decisions at the preregistered threshold 0.5 on **83/800 (10.4%)** questions for Qwen, **116/800 (14.5%)** for Shepherd, and **115/800 (14.4%)** for Skywork (Table 1). The separator-only change alone alters decisions on 40/800 (5.0%) and 51/800 (6.4%) questions for the marker interfaces, and direct S-P/S-U disagreement exceeds that comparison on the same questions by $+5.4$ and $+8.1$ percentage points (paired bootstrap intervals $[+2.9, +7.9]$ and $[+5.6, +10.6]$): a paired comparative rate difference, not an additive causal decomposition. For the readout-only interface no separator comparison exists, and its 14.4% is attributable by construction to readout selection (§6).

Disjoint panel. The disjoint panel (100 per subset, frozen before scoring, 400/400 scored) reproduces the effect: 44/400 (11.0%), 57/400 (14.3%), and 40/400 (10.0%), with the control patterns recurring; Skywork's point estimate is numerically lower. As an internal prospective replication (same benchmark, checkpoints, code, team) it is not independent confirmation, and the effect sizes are not claimed to be identical.

Robustness and its limits. Two qualifications apply. Threshold sensitivity differs across interfaces: 70 of 83 Qwen and 90 of 115 Skywork disagreements persist at every threshold in $[0.48, 0.52]$, against only 36 of 116 for Shepherd, whose disagreement mass is threshold-proximal. And disagreement remains when cut displacement is restricted to a local token neighborhood: a matched control that relocates each punctuation cut within a $\pm$2-token caliper on the canonical tokenizer, with unmatched cuts removed from both arms, gives 49/799 (6.1%), 104/799 (13.0%), and 97/799 (12.1%), so disagreement is not limited to widely separated scoring locations (Table 2); the observed incompatible pairs themselves falsify exact decision invariance on this panel. All headline descriptive rates are fp32-canonical; the largest bf16→fp32 shift is Shepherd's direct rate (13.0%→14.5%), and the individual certified Shepherd cases differ substantially between bf16 and fp32 while its aggregate mechanism channel is stable (construction, native-grid displacement, and precision detail: Appendix C).

5.2 Why the preregistered marginal contrast missed this

Two policies can each differ from a reference equally often without agreeing with each other. As an illustration, suppose each policy changes 10 of 100 reference decisions but on disjoint sets of questions: their reference-relative rates are equal, yet they disagree with each other on 20 questions. The

	Qwen	Shepherd	Skywork
Direct, original (800)	83	116	115
Wilson 95%	[8.4, 12.7]	[12.2, 17.1]	[12.1, 17.0]
Direct, disjoint (400)	44	57	40
Wilson 95%	[8.3, 14.4]	[11.2, 18.0]	[7.4, 13.3]
Persist $\pm$0.02 (of orig.)	70	36	90
at $\pm$0.01 / 0.05	77 / 49	67 / 11	101 / 46
Matched, original (799)	49 (6.1%)	104 (13.0%)	97 (12.1%)
Matched, disjoint (400)	26 (6.5%)	42 (10.5%)	37 (9.3%)
Sep.-only, disjoint (400)	16	32	n/a

Table 2: Robustness ladder at τ=0.5 (fp32), grouped as direct disagreement, threshold persistence, local-position controls, and the disjoint separator-only comparison; parenthesized labels give denominators. Wilson rows are simple binomial descriptive 95% intervals in percent. "Persist" rows count original-panel disagreements that persist at every threshold in the closed band (evaluated exactly over band endpoints and score breakpoints), not new whole-panel rates. bf16 sensitivity for the disjoint separator-only cells: 21/400 for Qwen and 40/400 for Shepherd.

preregistered primary tested the difference between the two reference-relative rates, not direct disagreement, so its null result does not establish that the policies make the same decisions. Concretely, the primary compared each policy's marginal displacement from F (exact McNemar, α=0.05, stratified 10,000-resample cluster-bootstrap intervals, gatekept Qwen→Shepherd) and was null: $\mathrm{RD} = -0.0150$, CI $[-0.0362, +0.0063]$, $p = 0.19$ on the locked bf16 basis; under the gate, Shepherd is descriptive ($+0.0013$) and Skywork exploratory (-0.0288). The interval rules out a 10-point marginal difference but not smaller effects. The gap between the two estimands is structural. F and split-view decisions are singleton released steps or BOT, so incompatibility is the discrete metric on outcomes:

Proposition 1 (triangle bound) *For views with singleton-or-*BOT *decisions, write* $m_V = \hat{d}(V, F)$ *and* $d = \hat{d}(S\text{-}P, S\text{-}U)$. *Then* $|m_{S\text{-}P} - m_{S\text{-}U}| \leq d \leq \min\{1, m_{S\text{-}P} + m_{S\text{-}U}\}$.

(Proof: Appendix A.) A near-zero marginal *difference* bounds direct disagreement only from below: two policies can displace equally often relative to F while displacing different questions, and on the fp32 rescore a 16-case marginal difference coexists with 116 direct disagreements for Shepherd. The lower bound is sharp, and the upper bound is sharp whenever $m_{S\text{-}P} + m_{S\text{-}U} \leq 1$.

6 Two Mechanisms of Policy Sensitivity

The previous section compared the punctuation and near-midpoint policies directly. This section changes the comparison: it compares each split input with the reference F while reading scores only at the original step endpoints, asking whether existing scores change enough to alter a decision; its counts do not partition the direct disagreement counts of Table 1. The interfaces expose two routes to sensitivity: marker-input policies change scores at the original endpoints (the score field moves), and the fixed-field interface changes decisions through the set of positions inspected (the readout set moves; Figure 2). The two marker-input views share marker type, count, treated steps, separators, and reasoning bytes, and the readout-only interface changes no input bytes at all, so the contrast is not explained by one arm receiving more marker perturbation.

Changing markers can change existing scores. Recomputing each split view's decisions at *only* the original endpoints isolates the input-side field change relative to F (it does not numerically decompose the direct S-P/S-U disagreements); Qwen shows 161 such common-endpoint decision changes under S-P (with 90 certified upward crossings), Shepherd 79 (12 certificates), and Skywork exactly 0, as the invariant requires (Table 3). A certificate is stronger than a common-endpoint change: when the earliest detected error moves later or disappears under an endpoint set that still contains the old decision endpoint, the old sub-threshold score must have risen to at least the threshold (Corollary 1); not every common-endpoint change is a certificate. Measured continuously over all shared endpoints, Qwen's score changes are small in the median but much larger in the upper decile (median 0.006, p90 0.21); Math-Shepherd's median absolute changes are small (0.013–0.015) while its disagreements are more threshold-sensitive (§5.1); Skywork's changes are exactly zero, verified on the stored rewards. The certificates prove the field changed at pre-existing endpoints without separating marker history from separators, retokenization, or position, so we call the mechanism input-side scoring-interface interference.

Changing readout positions can change decisions with fixed scores. Skywork is a mechanism control, not a competitive verifier at τ=0.5: the threshold approximately bisects its released-

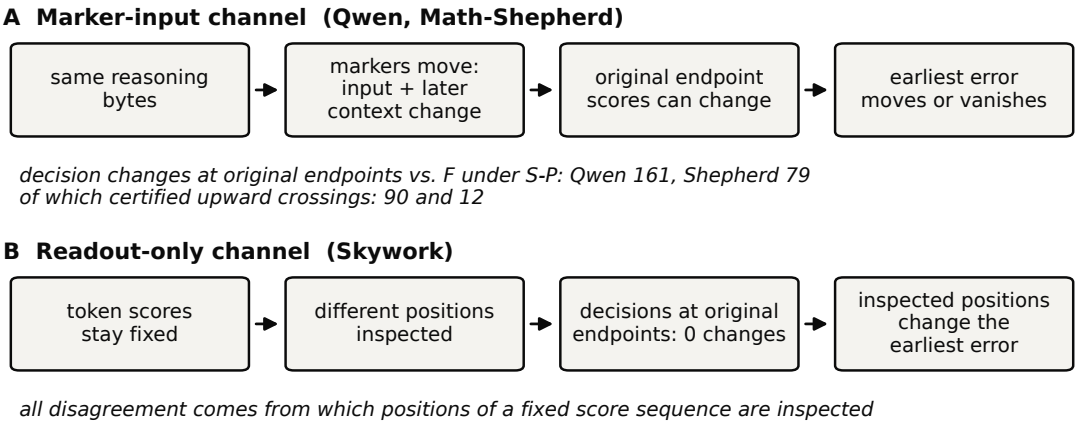


Figure 2: The two channels, each compared with the reference F. Marker-input policies (top) change the rendered input, so scores at the original endpoints can change: 161 (Qwen) and 79 (Shepherd) questions change their decision when the split-input rewards are read at only the original endpoints, of which 90 and 12 are certified upward crossings (Table 3). The readout-only policy (bottom) keeps one bitwise-fixed token-score sequence, so decisions at the original endpoints never change (exactly 0) and all disagreement comes from which positions are inspected.

	Changed vs. F		Certified	
Interface	S-P	S-U	S-P	S-U
Qwen	161	139	90	80
Shepherd	79	72	12	43
Skywork	0	0	0	0

Table 3: Reference-relative mechanism diagnostics (fp32, τ=0.5, original n=800 panel): numbers of questions whose decision, recomputed from the split-input rewards at only the original endpoints, differs from F ("Changed vs. F"), and the subset that are certified upward crossings. These are separate diagnostics, not additive parts of Table 1; Skywork's zeros are structural under the fixed-field realization.

step-end values (55.5% below τ against 20.3% and 17.9% for the marker interfaces). Its token scores are bitwise fixed, yet changing the set of inspected positions changes the earliest-error decision on 14.4% of questions: field invariance does not imply readout-set invariance, a property of the evaluated readout policy rather than a defect of token-level value modeling. The punctuation and near-midpoint policies do read different portions of the field (63.4% vs. 55.3% below threshold), but the mechanism claim requires only that the field is fixed and the actions differ. Direct disagreement remains 11.4% under a non-overshooting floor mapping, disfavoring a pure token-overshoot artifact (Appendix C).

7 Evaluation Consequence and Scope

Changing the scoring policy also changes which response is selected from a shared candidate pool. In *up-to-eight* common-candidate pools (identical candidates within each interface, ranked per policy by minimum block reward with ties broken by the lowest candidate index; 167 questions), the selected response differs on 104/167 (62.3%), 56/167 (33.5%), and 88/167 (52.7%) questions for Qwen, Shepherd, and Skywork; restricting to pools that retain all eight candidates gives 86/132, 45/132, and 70/131 (Table 4). This is selected-response disagreement, not evidence of an answer-accuracy change.

On the analyzed answer-defined subsets (n=64/62/64) we observe no question on which the selected response is correct under only one policy, so on those subsets the analysis establishes selection instability rather than an answer-quality change; pool composition, the margin ladder, the similarity heuristic, and Skywork's documented arithmetic-mean aggregation are in Appendix C. Across the reference and split policies, pooled ProcessBench-style harmonic F1 (0–1 scale) also varies by up to 0.046 with cohort- and interface-dependent sign, and clean-prefix retention moves with the policy (Appendix B).

8 Conclusion

An earliest-error prediction depends on the scoring-location policy as well as the PRM checkpoint.

	Qwen	Shepherd	Skywork
Selected response differs (167)	104	56	88
Full-eight pools only	86/132	45/132	70/131

Table 4: Selected-response disagreement under up-to-eight common-candidate selection (fp32, S-P vs. S-U, minimum aggregation, lowest-index tie rule). Margin, similarity, and answer analyses with their conditioning sets: Appendix C.

On the selected original panel, two count-matched policies over unchanged reasoning text disagree on roughly 10–15% of questions, and a prospectively frozen disjoint panel also shows disagreement. Marker-based policies can change scores at existing endpoints, while a readout-only policy can change decisions by inspecting different positions of a fixed score sequence. These findings support a concrete reporting requirement: evaluations should declare marker rendering, step separators, readout positions, the mapping to original steps, and the decision threshold, and compare models under matched policies; if several policies are relevant, their distribution and aggregation rule should be prespecified (Appendix D). The reported rates describe controlled sensitivity, not prevalence under naturally occurring scoring policies.

9 Limitations

These policies are controlled stress tests, not samples from deployment policies, so the rates are not prevalence estimates. The study covers a selected, jointly splittable ProcessBench cohort and three scalar interfaces; it does not establish generality to longer traces, other benchmarks, or current generative and reasoning-style verifiers (She et al., 2025; Zhao et al., 2026; Khalifa et al., 2026; Zhong et al., 2026), and score-scale reliability is a further active concern for scalar PRMs (Park et al., 2025). Reweighting to eligible-population subset frequencies (question keys or response rows) changes the headline rates by at most 1.6 percentage points (Appendix C; population scope: Appendix D).

The original direct comparison was post hoc; the prospectively frozen disjoint panel reproduced nonzero disagreement but shares the benchmark, checkpoints, code, and team. Threshold 0.5 is not calibrated across interfaces, Math-Shepherd's disagreements are threshold-proximal, and its individual certified cases differ substantially between bf16 and fp32 while its aggregate channel is stable (Appendix C).

The preregistration, amendments, deviation log, code, and raw rewards are not publicly released, so the chronology in Appendix D rests on the authors' internal records and the reproducibility record is the declared configuration in Table 5. The marked interventions are composite (readout set, input tokens, later context), and the sequential mechanism diagnostic is order-specific rather than a unique causal allocation. The selection analysis establishes selected-response instability, not answer-quality change, and the matched-coverage analysis is a failure to detect a residual difference, not a coverage attribution.

Ethics Statement

No private data, human subjects, or new annotation were used; the study evaluates model interfaces and is not human-performance or deployment-safety evidence.

References

Aakriti Agrawal, Souradip Chakraborty, Armin Saghafian, Nihal Sharma, Rizal Fathony, Nam H. Nguyen, C. Bayan Bruss, Amrit Singh Bedi, and Furong Huang. 2026. The hidden bias of process reward models: PRISM for rewarding the right reasoning. *arXiv preprint arXiv:2606.09078*.

Lang Cao, Renhong Chen, Yingtian Zou, Chao Peng, Huacong Xu, Yuxian Wang, Wu Ning, Qian Chen, Mofan Peng, Zijie Chen, Peishuo Su, and Yitong Li. 2025. More bang for the buck: Process reward modeling with entropy-driven uncertainty. *arXiv preprint arXiv:2503.22233*.

Hongzhan Chen, Tao Yang, Shiping Gao, Ruijun Chen, Xiaojun Quan, Hongtao Tian, and Ting Yao. 2025. Discriminative policy optimization for token-level reward models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 9546–9565. PMLR.

Alexey Dontsov, Anton Korznikov, Andrey V. Galichin, and Elena Tutubalina. 2026. Out of distribution, out of luck: Process rewards misguide reasoning models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 421–435, Rabat, Morocco. Association for Computational Linguistics.

Artyom Gadetsky, Maxim Kodryan, Siba Smarak Panigrahi, Hang Guo, and Maria Brbic. 2026. Unsupervised process reward models. *arXiv preprint arXiv:2605.10158*.

Jujie He, Tianwen Wei, Rui Yan, Jiacai Liu, Chaojie Wang, Yimeng Gan, Shiwen Tu, Chris Yuhao Liu,

Liang Zeng, Xiaokun Wang, Boyang Wang, Yongcong Li, Fuxiang Zhang, Jiacheng Xu, Bo An, Yang Liu, and Yahui Zhou. 2024. Skywork-o1 Open Series. Version 1.0.0. Model card: https://huggingface.co/Skywork/Skywork-o1-Open-PRM-Qwen-2.5-1.5B; checkpoint revision 98d69606595eedbdbbbf0a7d28efdcd462ba6a67, accessed 2026-07-30.

Yulan Hu, Sheng Ouyang, Jinman Zhao, and Yong Liu. 2025. Coarse-to-fine process reward modeling for mathematical reasoning. *arXiv preprint arXiv:2501.13622*.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2026. Process reward models that think. *Transactions on Machine Learning Research*.

Peng Kuang, Yanli Wang, Xiaoyu Han, Yaowenqi Liu, Kaidi Xu, and Haohan Wang. 2026. Optimal aggregation of LLM and PRM signals for efficient test-time scaling. In *The Fourteenth International Conference on Learning Representations (ICLR)*.

Jung Hyun Lee, June Yong Yang, Byeongho Heo, Dongyoon Han, Kyungsu Kim, Eunho Yang, and Kang Min Yoo. 2025. Token-supervised value models for enhancing mathematical problem-solving capabilities of large language models. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Yuliang Liu, Junjie Lu, Chaofeng Qu, Zhaoling Chen, Zefan Cai, Jason Klein Liu, Chonghan Liu, Yunhui Xia, Li Zhao, Jiang Bian, Chuheng Zhang, Wei Shen, and Zhouhan Lin. 2025. AdaptiveStep: Automatically dividing reasoning step through model confidence. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 39016–39031. PMLR.

Shrey Pandit, Austin Xu, Xuan-Phi Nguyen, Yifei Ming, Caiming Xiong, and Shafiq Joty. 2026. Hard2Verify: A step-level verification benchmark for open-ended frontier math. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22502–22517, San Diego, California, United States. Association for Computational Linguistics.

Young-Jin Park, Kristjan Greenewald, Kaveh Alim, Hao Wang, and Navid Azizan. 2025. Know what you don't know: Uncertainty calibration of process reward models. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*. ArXiv:2506.09338.

Qwen Team. 2024. Qwen2.5-Math-7B-Instruct. https://huggingface.co/Qwen/Qwen2.5-Math-7B-Instruct. Hugging Face model card; revision ef9926d75ab1d54532f6a30dd5e760355eb9aa4d, accessed 2026-07-30.

Qwen Team. 2025. Qwen2.5-Math-PRM-7B. https://huggingface.co/Qwen/Qwen2.5-Math-PRM-7B. Hugging Face model card; revision 0610740060112df12585d00a1c5f4624d2f59051, accessed 2026-07-30.

Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. 2025. R-PRM: Reasoning-driven process reward modeling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13438–13451, Suzhou, China. Association for Computational Linguistics.

Ibne Farabi Shihab, Fariya Afrin, Sanjeda Akter, and Anuj Sharma. 2026. EST-PRM: Stress-testing process reward models before they become load-bearing. *arXiv preprint arXiv:2606.00437*.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. PRMBench: A fine-grained and challenging benchmark for process-level reward models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25299–25346, Vienna, Austria. Association for Computational Linguistics.

Nishal Thomas and Noel Thomas. 2026. FormInv: A measurement protocol for semantic invariance in mathematical reasoning benchmarks. *arXiv preprint arXiv:2605.29001*.

Rishabh Tiwari, Aditya Tomar, Udbhav Bamba, Monishwaran Maheswaran, Heng Yang, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. 2026. Reward under attack: Analyzing the robustness and hackability of process reward models. *arXiv preprint arXiv:2603.06621*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.

Teng Wang, Zhangyi Jiang, Zhenqi He, Hailei Gong, Shenyang Tong, Wenhan Yang, Zeyu Li, Yanan Zheng, Zifan He, Zewen Ye, Shengjie Ma, and Jianping Zhang. 2026. Towards hierarchical multi-step reward models for enhanced reasoning in large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 565–576, San Diego, California, United States. Association for Computational Linguistics.

Zhaohui Yang, Chenghua He, Xiaowen Shi, Shihong Deng, Linjing Li, Qiyue Yin, and Daxin Jiang. 2025. Beyond the first error: Process reward models for reflective mathematical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 4711–4728, Suzhou, China. Association for Computational Linguistics.

Lingyin Zhang, Jun Gao, Xiaoxue Ren, and Ziqiang Cao. 2026. The bidirectional process reward model. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12564–12580, San Diego, California, United States. Association for Computational Linguistics.

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

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. 2026. GenPRM: Scaling test-time compute of process reward models via generative reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(41):34932–34940.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1009–1024, Vienna, Austria. Association for Computational Linguistics.

Congmin Zheng, Jiachen Zhu, Zhuoying Ou, Yuxiang Chen, Kangning Zhang, Rong Shan, Zeyu Zheng, Mengyue Yang, Jianghao Lin, Yong Yu, and Weinan Zhang. 2026. A comprehensive survey of process reward models: Data generation, model construction, and usage. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3591–3607, San Diego, California, United States. Association for Computational Linguistics.

Jianyuan Zhong, Zeju Li, Zhijian Xu, Xiangyu Wen, Kezhi Li, and Qiang Xu. 2026. Solve-Detect-Verify: Inference-time scaling with flexible generative verifier. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 47377–47401, San Diego, California, United States. Association for Computational Linguistics.

A Formal Results and Verification

Setting. Let $s_V(p) \in [0,1]$ be the score view V assigns to source endpoint p, endpoints ordered by source position; for a field s on an endpoint set B, let $d_s(B)$ be the earliest $p \in B$ with $s(p) < \tau$, else BOT.

Lemma 1 (fixed-field readout monotonicity) *Let $B \subseteq B'$ be finite subsets of a totally ordered source-coordinate domain and let fields s on B and s' on B' agree on the common set: $s'(p) = s(p)$ for every $p \in B$. Then $d_s(B) \neq$ BOT implies $d_{s'}(B') \neq$ BOT and $d_{s'}(B') \preceq d_s(B)$ in source order; and $d_s(B) =$ BOT implies $d_{s'}(B')$ is* BOT *or a new crossing in $B' \setminus B$.*

Corollary 1 (upward-crossing certificate) *With $B \subseteq B'$ as in Lemma 1 but s' not assumed to agree with s on B: if $d_s(B) = p \neq$ BOT while $d_{s'}(B')$ is* BOT *or strictly later than p in source order, then agreement fails at the old decision endpoint itself, $s'(p) \geq \tau > s(p)$.*

Proof of Lemma 1. If $d_s(B) = p$ then $s(p) < \tau$, and $p \in B'$ with $s'(p) = s(p) < \tau$ by agreement on B, so the earliest sub-threshold endpoint of B' under s' exists and is at or before p. If $d_s(B) =$ BOT, all of B is at or above τ under s and hence under s', so any crossing lies in $B' \setminus B$. □

Proof of Corollary 1. Suppose $d_s(B) = p \neq$ BOT, so $s(p) < \tau$. If $s'(p) < \tau$ held as well, then p would be a sub-threshold endpoint of B' under s' and $d_{s'}(B')$ would be at or before p, contradicting $d_{s'}(B') =$ BOT or $d_{s'}(B') \succ p$. Hence $s'(p) \geq \tau > s(p)$. □

Proof of Proposition 1. For split views and F, every decision is a singleton released step or BOT, and the frozen compatibility rule reduces to outcome inequality, so $I(V, W)$ is the discrete metric on the outcome space of released-step indices together with BOT. Write $X = I(S\text{-}P, F)$, $Y = I(S\text{-}U, F)$, $Z = I(S\text{-}P, S\text{-}U)$. Pointwise, $|X - Y| \leq Z \leq X + Y$ by the triangle and reverse-triangle inequalities for the discrete metric; then $|\mathbb{E}X - \mathbb{E}Y| \leq \mathbb{E}|X - Y| \leq \mathbb{E}Z$ and $\mathbb{E}Z \leq \mathbb{E}X + \mathbb{E}Y$. The restriction to singleton outcomes is essential: interval-overlap compatibility for wider decision intervals is reflexive and symmetric but not transitive, so the bound does not extend to it. □

Rendering and readout conventions. The punctuation and near-midpoint cuts require ≥8 canonical-tokenizer tokens per side and fall outside paired TEX; all split geometry is defined under the frozen Qwen2.5-Math-PRM-7B tokenizer and attributed back to Unicode code-point source coordinates under fail-closed coverage checks ("byte-identical" refers to the UTF-8 encoding covered by the pinned hashes). Math-Shepherd's tag is rendered space-prefixed per its model card; Skywork readouts use the token whose offset span contains each block's last character, because BPE merges block-final punctuation with following separators, and its release documents newline step-token readouts with arithmetic-mean Best-of-N aggregation (He et al., 2024), so our interior readouts are off-reference stress tests of the same value field.

Verification invariants. Fail-closed checks enforce exact-once character coverage, monotone intervals, marker/block counts, pinned input hashes, identical marker-stripped text (R0 excepted, whose separator bytes differ by design), and reconstruction of every decision from raw rewards; 800 rows × 5 views (F, R0, S-P, S-U, and M0, an even-adjacent-merge control view whose analyses are not reported here) were byte-verified per interface with zero interface-specific cuts, and no input was truncated. Every row has equal S-P/S-U unique token-readout counts with zero collisions under both readout mappings. Complete revisions, hashes, and the full deviation record are retained in the authors' internal archive, which is not publicly released; the executed conventions are declared in Table 5.

B Evaluation Consequence Details

Metrics. ProcessBench-style harmonic F1 is the harmonic mean of two class accuracies: on gold-error trajectories a prediction is correct iff its released step equals the annotated first error, and on all-correct trajectories iff it is BOT; the value is undefined when a class is empty. Macro F1 averages the per-subset values over GSM8K, MATH, OlympiadBench, and Omni-MATH; pooled F1 computes the two class accuracies over the pooled selected cohort. The two aggregate differently and need not agree. Gold-error retention is the fraction of gold-error rows whose accepted clean prefix (the released steps strictly before the flagged block; the whole response under BOT) still contains the annotated first error, so larger values are worse.

Item	Declaration, as executed
Checkpoints	Hugging Face repositories at the pinned commit revisions: `Qwen/Qwen2.5-Math-PRM-7B` at `0610740060112df12585d00a1c5f4624d2f59051`; `peiyi9979/math-shepherd-mistral-7b-prm` at `45dc0a3c9ec699b645085c098ed38dc99fba4617`; `Skywork/Skywork-o1-Open-PRM-Qwen-2.5-1.5B` at `98d69606595eedbdbbbf0a7d28efdcd462ba6a67`; candidate generator `Qwen/Qwen2.5-Math-7B-Instruct` at `ef9926d75ab1d54532f6a30dd5e760355eb9aa4d`. Tokenizers are loaded at the same revisions.
Qwen input	Pinned chat template with system prompt "Please reason step by step, and put your final answer within \boxed{}.", user turn = problem, assistant turn = rendered response (no generation prompt). `<extra_0>` immediately after every visible block; F glue `\n\n` after each non-final marker at a released-step edge and no glue at a within-step cut; R0 = markers only, no glue (model-card rendering). Reward = softmax positive-class probability at each marker position.
Shepherd input	Plain text `problem + " " + rendered response`. Tag = one space followed by the two Cyrillic letters U+043A U+0438 (the model card's "ki" tag; space-prefixed, tag token id 12902) immediately after every block; F glue `\n\n`, R0 glue `\n` (model-card newline-separated steps). Reward = $P(+)/(P(+)+P(-))$ from the logits of token ids 648 (+) and 387 (-) at each tag position.
Skywork input	One string per row: `<|im_start|>` + problem + `\n` + response + `\n`, response blocks joined by the original `\n\n`; no markers; the full string is tokenized once with offset mapping. Value = sigmoid of the per-token value head. Containing mapping: the token whose offset span contains each block's last character; floor mapping: the last token whose span ends at or before it.
S-P cut	After `.`, `?`, or `!` immediately followed by whitespace; outside balanced TEX spans; $\geq$8 pinned Qwen2.5-Math-PRM-7B tokenizer tokens on each side; minimize token imbalance, lowest character offset on ties; the whitespace belongs to the right child.
S-U cut	A whitespace run outside balanced TEX, neither preceded nor followed by `.?!`; $\geq$8 tokens per side; minimize $\lvert\text{left}-\text{right}\rvert$ tokens (token-midpoint rule), earliest offset on ties. A step is split iff both cuts exist (joint admissibility), exactly one cut per such step, so S-P and S-U have identical split-step sets and marker counts. Matched arm: each S-P cut relocated to the nearest admissible non-punctuation whitespace within $\pm$2 canonical tokens.
Cohort gates	E1 valid label and 4–12 nonempty released steps; E2 R0 full-chat rendering $\leq$2,048 pinned-tokenizer tokens; E3 at least one jointly admissible step; 26 pilot-contaminated rows removed; one row per question key; 200 keys per subset by a pre-committed salted SHA-256 key over the subset, question key, and row id. Context limit 4,096 tokens; no input truncated.
Decision rule	Earliest visible block with reward < 0.5, else BOT; two decisions are incompatible iff BOT status differs or both are non-BOT with disjoint inclusive released-step intervals.
Numerics	fp32 canonical rescore with bf16 as a sensitivity; `transformers` 4.48.3, Python 3.10.20, one NVIDIA RTX A6000; the rendered-text SHA-256 of every row is asserted equal across passes.

Table 5: Interface declaration: the rendering, readout, cut, and selection conventions needed to reproduce the scored inputs, i.e., the quantities this paper asks PRM evaluations to report.

Implementation sanity check. Macro F1 of the F view is 69.9 (original panel) and 76.7 (disjoint panel) for Qwen2.5-Math-PRM against a published 73.5, and 28.8 and 29.3 for Math-Shepherd against a published 31.5 (Zheng et al., 2025; Zhang et al., 2025): within about four F1 points in every cell, a coarse check on the scoring-interface implementation only.

Panel	Interface	F	S-P	S-U
Original	Qwen	0.699	0.673	0.693
Original	Shepherd	0.281	0.298	0.261
Original	Skywork	0.336	0.322	0.295
Disjoint	Qwen	0.772	0.726	0.732
Disjoint	Shepherd	0.279	0.325	0.289
Disjoint	Skywork	0.361	0.339	0.342

Table 6: Pooled ProcessBench-style harmonic F1 (0–1 scale; fp32; τ=0.5; post hoc, not leaderboard-comparable) by panel, interface, and view.

Pooled harmonic F1. Across the reference and split views (Table 6) the pooled value varies by up to 0.046 (disjoint Qwen, F vs. S-P), with cohort- and interface-dependent sign; the largest direct S-P vs. S-U difference is 0.037 (original Shepherd).

Clean-prefix retention. In a clean-prefix use (accept the released steps strictly before the flagged error), gold-error retention (larger is worse) on the 504 gold-error rows moves with the policy at τ=0.5: 0.276/0.361/0.321 for Qwen under F/S-P/S-U, 0.742/0.714/0.760 for Shepherd, and 0.145/0.115/0.113 for Skywork. These are operating-point differences. At matched accepted-character coverage the S-P$-$S-U gold-error-retention gap is -0.008 $[-0.023, +0.009]$ (Qwen), -0.003 $[-0.015, +0.009]$ (Shepherd), and -0.002 $[-0.015, +0.010]$ (Skywork), replicating on the disjoint panel with every interval straddling zero. The comparison interpolates re-

tention piecewise-linearly in coverage between adjacent threshold operating points and evaluates both policies at the midpoint of their own τ=0.5 coverages inside every bootstrap replicate; it is a failure to detect a residual difference under a post hoc randomized-threshold interpolation, not a coverage attribution.

C Core Robustness

Thresholds beyond the band. Under post hoc cross-panel calibration, one threshold per interface is chosen on the disjoint panel's F view as the achievable all-correct false-positive rate nearest 0.10 (the rate is a step function of τ; ties are broken by the smaller count) and applied unchanged to the original panel, where direct disagreement stays high: 75/99/114 of 800. This is a post hoc sensitivity, not a further prospective confirmation. Among the τ=0.5 direct disagreements on the original panel (83/116/115), exactly one policy is strictly correct in 52/36/65 cases, where a decision is strictly correct if its released step equals the annotated first-error step or if it is BOT on an all-correct trajectory; policy choice therefore changes correctness, not just labels. The nine-point trapezoidal summary of direct disagreement over $\tau \in \{0.1, \ldots, 0.9\}$ is 0.126/0.117/0.124, an unnormalized integral over a τ-range of width 0.8 (divide by 0.8 for the average rate) and a grid summary rather than the exact breakpoint integral. Reweighting the equal-subset rates to eligible-population subset frequencies moves each headline rate by at most 1.6 percentage points (eligible question keys: 10.4→11.1%, 14.5→15.7%, 14.4→13.5%; eligible response rows: 11.4%, 16.1%, 13.5%).

Matched near-position control. The near-position arm relocates each matched punctuation cut to the nearest admissible non-punctuation whitespace within a $\pm$2 left-token caliper (3,108 of 3,206 cuts on 799/800 rows), with unmatched cuts removed from *both* arms under a fail-closed row-level equal-count assertion (this corrects an earlier construction whose arms were not marker-count matched). The corrected contrast is 49/799, 6.1% [4.7, 8.0] (Qwen); 104/799, 13.0% [10.9, 15.5] (Shepherd); 97/799, 12.1% [10.1, 14.6] (Skywork). Relocation is token-local on every native grid (maximum 2 Skywork tokens; maximum 5 Mistral tokens, 99.7% within 3). The exploratory one-cut variant relocates a single hash-selected cut by one to two canonical tokens, retaining all other scoring locations, and still changes 7/400 (1.8%), 17/400 (4.3%), and 20/400 (5.0%) of disjoint-panel decisions. Paired separator-vs-placement differences use a subset-stratified question-level paired bootstrap (10,000 resamples; discordant pairs 74/31 and 85/20).

Precision sensitivity. A later frozen amendment rescored both panels at fp32, the canonical basis; bf16 is retained as a sensitivity. Reward differences $|r_{\mathrm{fp32}} - r_{\mathrm{bf16}}|$ have medians 0.0007 (Qwen), 0.0067 (Shepherd), and 0.0026 (Skywork); the largest headline shift is Shepherd's direct rate (13.0%→14.5%), the other five direct headline rates move $\leq$1.2pp, and whether a majority of Shepherd's direct disagreements persisted across the threshold band changed under fp32 (36/116 survive $\pm$0.02, from 57/104 at bf16). Qwen and Skywork preserve most disagreement identities across precisions; for Shepherd the individual certified cases differ substantially between bf16 and fp32 while its aggregate common-endpoint count is stable (79 vs. 78 for S-P), which is why the Shepherd mechanism claim rests on the aggregate channel.

Field, drift, and mapping detail. Over all 5,343 shared endpoints, Qwen's S-P drift has median 0.006, p90 0.21, 18.5% above 0.10 (Spearman 0.96); Shepherd's medians are 0.015/0.013 with at most 1.0% above 0.10 (Spearman 0.992); Skywork's drift is exactly 0 at every common endpoint (Spearman 1.0). Among gold-error rows where F fires at τ=0.5 (437 rows), the mean signed change at F's predicted crossing is $+0.175$ (Qwen S-P) and $+0.130$ (S-U), consistent with erasure of existing F flags; Shepherd's is near zero (-0.004, $+0.017$; 277 rows); this signed statistic is threshold-conditioned and selection-prone, so the absolute drift above is the threshold-free evidence. The threshold approximately bisects Skywork's released-step-end values (55.5% below τ=0.5 vs. 20.3% and 17.9% for the marker interfaces), its punctuation cuts read 8.1 points more below-threshold field than its midpoint cuts (63.4% vs. 55.3%), and under the preregistered non-overshooting floor mapping direct disagreement is 91/800 (11.4%) against 115/800 (14.4%) under the containing mapping, with the corrected matched rate 85/799 (10.6%) against 97/799 (12.1%); both ratios exceed the pre-committed 0.5 persistence bar.

Up-to-eight selection diagnostics. Eight candidates per question were generated once with `Qwen/Qwen2.5-Math-7B-Instruct` (Qwen Team, 2024) (temperature 0.8, top-p 0.95, at most 1,024 new tokens, per-candidate seeds fixed before generation). The question population for this generation is different from the released-response cohort: it is the set of distinct ProcessBench GSM8K and MATH problems excluding the original and disjoint panels and the pilot rows, with no released-response gates (E1–E3 apply to released responses, not to newly generated candidates). This leaves 69 GSM8K problems, all of which were taken, and a larger MATH pool from which 100 were drawn by salted hash, for 169 questions; two MATH questions had no candidate scored under both policies of a contrast (the same two for every interface) and are excluded, leaving the 167 analyzed pools. Pools are policy-matched within each interface: a question's common pool is the set of candidates scored under both policies of the contrast, candidates scored under only one policy are dropped from both rankings, and pools are therefore up to eight (eight candidates on 132 questions, 131 for Skywork, and as few as one) and are not candidate-identical across interfaces. Each policy selects the candidate with the highest aggregated score (minimum block reward for the headline; log-product and geometric mean as sensitivities), ties broken by the lowest candidate index; single-candidate pools cannot disagree, count as agreement in the 167-question rate, and are excluded from margin analyses. Selected-response disagreement conditional on both policies' top-two margins strictly exceeding a rung (Table 7) is monotone non-increasing in every interface under all three aggregations; margins are within-interface quantities and are not comparable across interfaces.

Rung $>$	0.001	0.005	0.01	0.02
Qwen, % (n)	42.6 (61)	27.0 (37)	19.4 (31)	12.5 (24)
Shepherd, % (n)	27.3 (143)	17.8 (107)	9.1 (88)	4.9 (61)
Skywork, % (n)	52.2 (136)	50.0 (110)	44.2 (86)	37.7 (61)

Table 7: Selected-response disagreement (%) among questions whose top-two raw score margins under *both* policies strictly exceed the rung, with the retained n in parentheses (minimum aggregation, fp32; rungs in reward units).

Trace similarity is the `difflib.SequenceMatcher` ratio on whitespace-normalized strings. Each disagreement pair is assigned to exactly one bucket by the frozen priority identical normalized text $>$ answer unparseable $>$ different parsed answers $>$ near-duplicate (same parsed answer, ratio $\geq$0.9) $>$ textually distinct, giving textually distinct / near-duplicate / different parsed answer / unparseable counts of 95/5/2/2 (Qwen), 46/5/0/5 (Shepherd), and 79/2/2/5 (Skywork) with no identical-text pairs; "textually distinct" is a declared string heuristic, not semantic distinctness. Answers are the last parseable `\boxed{}` payload after whitespace normalization; the answer-defined subset requires both selected answers to parse and a gold answer to be available (n=64/62/64), and the answer-diverse subset requires at least two distinct parsed answers in the pool. On answer-diverse pools selected-answer disagreement is 2/27, 0/22, and 2/24; on the answer-defined subsets the paired correctness deltas are exactly 0 with 0 discordant pairs (exact one-sided 95% upper bounds $1 - 0.05^{1/n}$ on the discordance probability: 0.046/0.047/0.046; 0.044 under the n=66 unparseable-as-incorrect sensitivity); zero observed discordance bounds, and does not exclude, a nonzero population rate. Under the documented Skywork reference aggregation (arithmetic mean over step readouts (He et al., 2024)), computed by a procedure that first reproduces the canonical minimum-aggregation counts exactly, selected-response disagreement is 103/167 (61.7%), 43/167 (25.7%), and 87/167 (52.1%): Skywork's rate under its own documented aggregation is nearly identical to the minimum-aggregation stress test (52.1% vs. 52.7%), and aggregation choice is an active evaluation variable even at the response level (Kuang et al., 2026).

D Epistemic Status and Scope

Amendments and deviations. Seven frozen amendments extended the preregistration (controls, disjoint panel, ranking analysis, floor mapping, fp32 rescore), each specified before its own scoring. Six deviations are logged with timestamps in the authors' internal, unreleased record; the three affecting analyses reported here are: Shepherd's tag rendering corrected to the model-card convention before any reward analysis, Skywork readouts mapped to the containing token before any reward was observed, and the first matched arms, which were not marker-count matched and are superseded by the corrected arms reported here.

Analysis	Status
Marginal S-P vs. S-U primary	Preregistered before scoring; null; locked to bf16
Direct disagreement	Post hoc on the original panel; prospectively estimated on the frozen disjoint panel
Matched, one-cut, band, floor controls	Frozen before their scoring runs
Precision rescore (fp32 canonical)	Preregistered full-cohort rescore; changed conclusions reported as changed
Downstream analyses (F1, coverage, selection)	Exploratory or post hoc; descriptive uncertainty

Table 8: Condensed epistemic status. Post hoc p-values are unadjusted unless stated; bootstrap intervals target repeated equal-subset draws from the four eligible strata, and eligible-population reweighting is reported separately (Appendix C).

Policy marginalization objects. For a declared, outcome-independent policy distribution μ and a per-example loss L, a marginalized benchmark reports $R_\mu(\theta) = \mathbb{E}_{(x,y)}\,\mathbb{E}_{\pi\sim\mu}\,L(D_{\theta,\pi}(x), y)$; for confusion-count metrics such as harmonic F1, per-policy averaging $\mathbb{E}_\mu[H(C_{\theta,\pi})]$ and pooled counting $H(\mathbb{E}_\mu[C_{\theta,\pi}])$ are different estimands, and the policy-draw mechanism (per benchmark run or per question) must be declared.

External validity. Selection funnel: of the 3,400 ProcessBench rows, 2,916 have 4–12 nonempty released steps (E1), 2,915 of those also render within the 2,048-token R0 ceiling (E2), and 2,842 also contain at least one jointly admissible step (E3), so the intervention is defined for 83.6% (2,842/3,400) of rows; removing 26 pilot-contaminated rows leaves the 2,816-row eligible response funnel (340/795/826/855 per subset), which collapses to 2,416 eligible question keys (321/763/511/821) under one row per key, from which the fixed panel draws 200 keys per subset by pre-committed salted hash. Reweighting to eligible-key or eligible-row frequencies is a sensitivity, not the sampling target. Selected and eligible-but-excluded rows differ modestly on score-blind features (all $|\text{SMD}| \leq 0.20$, largest on length), with selected rows somewhat shorter and slightly less error-heavy; the SMD audit uses a local proxy of the E3 gate (the authoritative gate needs the pinned tokenizer): the proxy admits 2,781 rows (331/779/824/847 per subset against the authoritative 340/795/826/855), 792 of the 800 selected rows pass it, and the comparison is between all 800 selected rows and the 1,989 proxy-admitted rows that were not selected. The reported rates are not estimates of full-ProcessBench or naturalistic-segmenter prevalence.

Embedding-Based Educational Taxonomies

Kia Karbasi[1], **Kaveh Karbasi**[2*], **Gregory Pottie**[1]
[1]University of California, Los Angeles [2]Tempus AI, Inc.

Abstract

Educational taxonomies organize educational content, such as exercise problems, into concepts, which are also called skills or knowledge components. These taxonomies define learning objectives in educational applications such as intelligent tutoring systems. Developed through human expertise and iterative refinement, they also provide fundamental structure for curricula, assessment, and models of student learning. Recent advances in semantic text embeddings make it possible to construct alternative educational taxonomies computationally from the content itself. In this paper, we investigate this possibility by clustering embeddings of math questions to construct embedding-based educational taxonomies. We compare these taxonomies with the existing Common Core State Standards (CCSS)-based knowledge-component taxonomies in two ASSISTments datasets and evaluate their utility in Deep Knowledge Tracing, a downstream task that relies on taxonomic mappings of questions to knowledge components. The embedding-based taxonomies exhibit substantial but incomplete correspondence with the existing taxonomies, revealing both preserved and alternative organizations of educational content. When used for knowledge tracing, they improve predictive performance by approximately 3–5 AUC percentage points. These results demonstrate the potential of semantic embeddings for automatically constructing educational taxonomies that both recover structure present in human-developed taxonomies and provide useful alternative representations for downstream educational modeling.

1 Introduction

Educational taxonomies organize educational content, such as exercise problems, into concepts. In educational data and modeling, these concepts are often called skills or knowledge components (KCs). A taxonomy specifies which questions are treated as instances of related knowledge, giving educators and computer systems a common way to describe content and learning objectives. These mappings provide structure for curricula, assessment, educational applications, and models of student learning (Koedinger et al., 2012).

Educational taxonomies are generally developed through human expertise and iterative refinement. The Common Core State Standards, for example, were developed through research, educator and expert input, comparison with existing standards, and public feedback (Common Core State Standards Initiative, n.d.). Such systems encode decisions about which distinctions matter for curricula and assessment, while KC models use those distinctions to relate educational tasks, instruction, and student learning (Porter et al., 2011; Koedinger et al., 2012). They are therefore useful educational frameworks, not direct evidence that knowledge is organized in one uniquely correct way.

These taxonomies also provide the categories that educational technologies use to organize content and model student learning. Intelligent tutoring systems connect educational materials to the concepts they are intended to teach or assess (Koedinger et al., 2012; Feng et al., 2009). Knowledge tracing uses sequences of questions and responses, along with the concepts assigned to those questions, to estimate student knowledge and predict later performance (Corbett and Anderson, 1995; Abdelrahman et al., 2023). Deep Knowledge Tracing (DKT) represents each response with its assigned KC, so earlier performance on a KC can inform later predictions (Piech et al., 2015). Systems that recommend exercises similarly use concept labels and estimates of student knowledge to select individualized practice (Ai et al., 2019).

Recent advances in text embeddings make it possible to organize educational content using patterns

[*]Tempus AI, Inc. had no role in the planning, execution, funding, or review of this work.

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 268–284
October 29, 2026 ©2026 Association for Computational Linguistics

in the content itself. Embeddings represent question text in a numerical space where similarities in language and meaning can be compared (Reimers and Gurevych, 2019). Clustering these representations groups nearby questions and assigns each question to one category (Matsuda et al., 2022). In this paper, we use semantic embeddings of math question text to induce embedding-based taxonomies. Here, a taxonomy refers to a mapping between content or questions and content categories or knowledge components. An induced taxonomy is one constructed from text embeddings. We compare these induced taxonomies with the original expert-developed taxonomies and test their utility in a downstream educational task that relies on question-to-KC mappings.

We study questions with available text from ASSISTments 2012 and 2017, two releases from a widely used benchmark family in knowledge tracing (Feng et al., 2009; Prihar et al., 2022; Abdelrahman et al., 2023). We embed the question text, reduce the representations with principal component analysis (PCA), and cluster them with k-means to construct taxonomies from embeddings. We first compare each taxonomy built from embeddings with the original KC taxonomy supplied with the dataset, using several measures of how the questions are grouped, along with category counts. We then replace the original mapping from questions to KCs with the induced mapping and compare DKT performance using the area under the receiver operating characteristic curve (AUC). Within each dataset, we use the same student-response records, sequence order, student split, DKT model family, training procedure, Bayesian hyperparameter search, and trial budget. Because an induced taxonomy may use a different number of categories, the number of KC labels and the number of model parameters tied to those labels may also differ. We tune DKT separately for each taxonomy, so the selected DKT hyperparameters may differ. The code and supplementary materials are available in a public GitHub repository.[1]

We ask three research questions:

RQ1: How closely do the induced and original KC taxonomies correspond?

RQ2: How does correspondence change as the induced taxonomy uses more categories?

RQ3: Under which taxonomy does DKT achieve higher AUC: the original KC taxonomy or the embedding-induced taxonomy?

[1] https://github.com/DKT-PROJECTS-UCLA/KC_Clusters

2 Related Work

2.1 Educational Taxonomies and Knowledge Components

KC models connect performance on related tasks to proposed units of knowledge used in teaching and models of student learning (Koedinger et al., 2012). A mapping from questions to KCs therefore represents a hypothesis about the knowledge required by each question. These models are designed for educational use; they are not direct measurements of how students think. Learning Factors Analysis shows that student data can be used to evaluate and refine an expert KC model (Cen et al., 2006). Whereas that approach compares possible refinements using student performance, we construct alternative categories from question content. We therefore use the existing KC taxonomy as a valuable reference, not as the only possible way to group the questions. Expert-developed taxonomies reflect instructional organization; induced taxonomies offer complementary groupings derived from question text. Whether these groupings align with how students organize or learn that content remains an open question.

2.2 Embedding-Based Clustering and Automated KC Discovery

Sentence embeddings represent text in a space that supports semantic comparison and clustering (Reimers and Gurevych, 2019). Text similarity can suggest shared content, but it does not by itself establish a common solution method or role in teaching. SMART uses k-means to cluster representations of instructional and assessment text in which terms are weighted by their frequency within a document and rarity across documents (TF–IDF), compares the induced categories with expert categories, and evaluates the resulting mapping with an Additive Factors Model (AFM) (Matsuda et al., 2022). KCluster uses language-model-based question similarity and affinity propagation to derive KCs, compares them with an expert taxonomy, and also evaluates them with AFM (Wei et al., 2025). These are the clustering studies most closely related to our work, although neither uses ASSISTments or DKT. Moon et al. extract KCs from question text and images, group them using embeddings, and evaluate the resulting categories with several

knowledge-tracing models, including DKT (Moon et al., 2025).

Together, these studies show that educational categories can be constructed from course and question content. Our study builds on this work by comparing induced and original taxonomies across every tested combination of PCA dimension and number of induced categories (K), then evaluating the induced mappings in DKT. We also ask whether settings where the taxonomies match more closely tend to produce higher AUC. The core methodology was developed independently of SMART and KCluster, which we identified as closely related approaches during subsequent literature review.

2.3 Knowledge Tracing and Text-Aware Student Modeling

Knowledge tracing uses question–response histories to model how student knowledge develops over time (Corbett and Anderson, 1995; Abdelrahman et al., 2023). DKT uses a recurrent neural network to predict later responses from earlier responses and their KC labels (Piech et al., 2015). For example, if two questions share a KC label, DKT can use a student's earlier performance on that KC when predicting a later response; under our method, the same process uses the induced KC labels instead. Methods that use question text add question representations to student models while retaining the existing skill labels (Su et al., 2018; Liu et al., 2021). Our approach uses embeddings before DKT is trained: the embeddings define an alternative mapping from questions to categories, and DKT is then trained with the induced labels. This lets us evaluate the mapping in two ways: how closely it matches the original taxonomy and the test AUC achieved by DKT with the induced labels.

3 Method

Figure 1 illustrates knowledge tracing with an induced taxonomy and the corresponding relabeling used for DKT evaluation. We first derive a mapping from questions to categories that has one level, using question text alone. We then replace the KC mapping supplied with the dataset with the induced mapping in DKT while keeping the records of student answers and their order unchanged.

3.1 Data and Building the Taxonomies

We use ASSISTments 2012 and 2017, two releases from a widely used benchmark family in knowledge tracing (Feng et al., 2009; Prihar et al., 2022; Abdelrahman et al., 2023), because their student interactions can be linked to question text from the ASSISTments Problem Bodies (PB) resource (ASSISTments Project, n.d.). The full 2012 release contains 179,999 questions and 265 KCs, of which 32,792 questions and 162 KCs belong to the PB text-linked subset. The full 2017 release contains 3,162 questions and 102 KCs, of which 638 questions and 68 KCs belong to the PB text-linked subset. These PB subsets are the text-bearing questions and KCs used for our text-based analyses. The largest category in the original 2017 taxonomy, noskill, is a non-substantive placeholder covering 142 of the 638 questions (22%). We exclude ASSISTments 2009 because its archived release is sensitive to preprocessing choices (Xiong et al., 2016). We refer to the dataset-provided question-to-KC mapping in each release as the original taxonomy. Table 1 summarizes the dataset sizes, the category counts tested, and the PCA and K setting reported for each dataset in the taxonomy comparison and DKT results.

We convert the HTML in each `problem_body` to plain text, retain image descriptions, remove scripts and style information, and normalize whitespace. We construct each induced taxonomy from these cleaned question texts. The original KC labels and student responses are not used during construction. Two models embed each question, PCA reduces the vectors, and k-means assigns each question to the nearest cluster center. We evaluate eight PCA dimensions and six values of K, the number of induced categories, for each dataset. The tested grids (2012: 50–300; 2017: 20–150) span values below, near, and above the original PB KC counts of 162 and 68, respectively. Each taxonomy assigns one label per question and has one level, not a hierarchy. The taxonomy is constructed from the complete set of linked question texts before splitting students. Following the standard evaluation for knowledge tracing implemented in pyKT (Liu et al., 2022), we split the data by student and evaluate students held out from training. The test set therefore contains students not seen during training rather than previously unseen questions.

Appendix A documents the permitted examples of question text, and Appendix B gives the exact embedding models, PCA dimensions, values of K, clustering settings, and procedure used to select DKT settings.

Problem Bodies were accessed under the ASSISTments terms of use for the data. The embed-

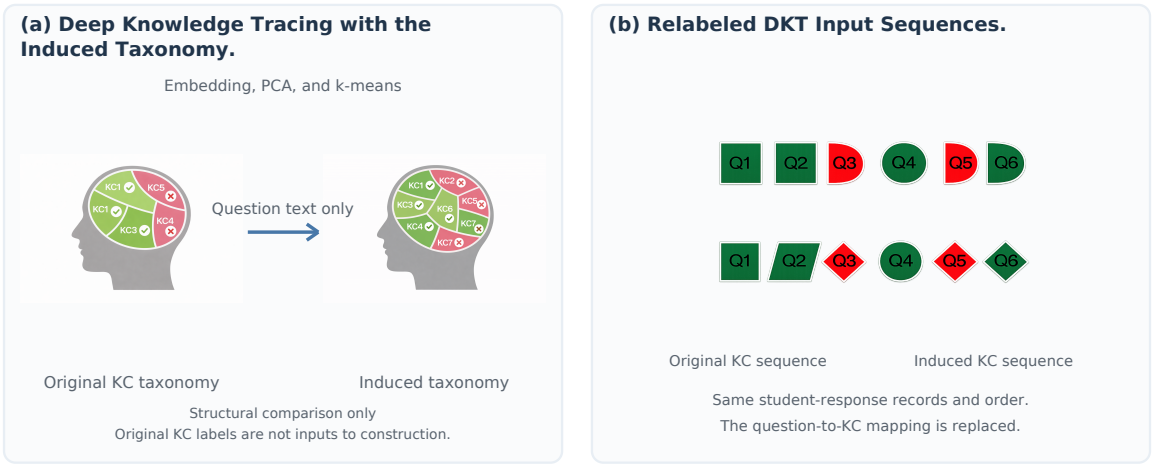


Figure 1: Knowledge tracing with an induced taxonomy and sequence relabeling. Panel (a) conceptually illustrates DKT tracking induced KCs; the illustrated mastery states are conceptual and do not represent measured student knowledge. Panel (b) shows the actual relabeling procedure for DKT evaluation: the original KC label in each interaction is replaced with the induced label while questions, responses, sequence order, student splits, and evaluation remain unchanged.

Dataset	Total questions	PB questions	Total KCs	PB KCs	Students	Responses
ASSISTments 2012	179,999	32,792	265	162	23,845	978,271
ASSISTments 2017	3,162	638	102	68	1,703	237,092

Table 1: Dataset and taxonomy summary. Total counts describe the full releases; PB counts describe the Problem Bodies text-linked question/KC subsets used for text-based analyses. Student and response counts describe the text-linked subset used for DKT evaluation. For both datasets, we construct induced taxonomies using outputs from OpenAI's `text-embedding-3-small` and `text-embedding-3-large`, shortened through the API's `dimensions` parameter to 768 and 1,536 dimensions, respectively (OpenAI, 2026); we use the PCA dimensions $\{10, 20, 40, 60, 80, 100, 120, 160\}$, with K values $\{50, 100, 150, 200, 250, 300\}$ for 2012 and $\{20, 40, 60, 80, 100, 150\}$ for 2017. PCA and k-means use a random seed of 42; k-means uses 10 initializations and at most 300 iterations. The reported (PCA, K) settings in Table 2 are (100, 300) for 2012 and (10, 100) for 2017. The PCA dimension and K are fixed before each DKT Bayesian search and are not selected by it.

ding inputs consisted only of cleaned text from the `problem_body` field and excluded grade information, original KC labels, and records of student answers. Because the access terms prohibit redistribution of the questions, neither the paper nor the public GitHub repository reproduces the original question text.

3.2 Taxonomy Comparison

We compare questions that have both an original and an induced label using the measures summarized in Appendix Table 3. Homogeneity measures whether each induced category primarily contains questions from one original category, while completeness measures whether each original category remains concentrated within one induced category. The remaining measures ask how similarly the two taxonomies group questions, whether pairs of questions stay together, and how different the groupings are overall. Because some scores change simply when the induced taxonomy uses more categories, we interpret them together rather than treating any one score as a complete evaluation. We report induced category sizes separately. For the reported 2017 DKT setting, Appendix Figure 5 shows how the 25 largest original categories are distributed across the induced taxonomy. Together, these measures describe how categories stay together, split, or mix across the two taxonomies. Educational usefulness requires separate evidence.

3.3 DKT Evaluation

We use DKT because it can use a student's performance on one KC when predicting later answers (Piech et al., 2015). KC labels are part of each sequence given to DKT, so replacing the origi-

nal labels tests how DKT performs with the induced taxonomy as a whole. We keep every question, response, and sequence position fixed and replace the original mapping from questions to KCs with the corresponding induced mapping. pyKT maps the resulting labels to consecutive concept identifiers (Liu et al., 2022). Varying K changes the number of labels and can also change the number of model parameters tied to those labels. Figure 1b illustrates this relabeling.

We run a separate Bayesian search for the original taxonomy and for each induced taxonomy defined by an embedding model, PCA dimension, and K. Within each dataset, all searches use the same DKT model family, candidate settings, number of trials, student split, training procedure, and selection rule. We reserve 20% of students for final testing, giving an approximate 64/16/20 training, validation, and test split. Within each search, we select the completed run with the highest validation AUC and report its test AUC. The embedding model, PCA dimension, and K are fixed before the search. Testing both embedding models across eight PCA dimensions and six values of K yields 96 induced taxonomies per dataset. Table 2 reports the highest test AUC across the induced taxonomies and the embedding model, PCA dimension, and K that produced it; Figure 4 shows the grid of PCA dimensions and K values for that embedding model. Appendix B gives the exact search spaces and training settings.

To test whether closer agreement between the taxonomies is related to higher AUC, we analyze the 48 settings of PCA dimension and K in each dataset. At each setting, we average each structural score across the two embedding models and separately average the two test AUC values. We compute Spearman correlations, then repeat the analysis after removing rank patterns linked to PCA dimension and K. We apply the Benjamini–Hochberg correction separately to each set of 15 comparisons (Benjamini and Hochberg, 1995). Appendix D reports the full procedure and coefficients; the analysis shows associations, not causal effects.

4 Results

4.1 RQ1: Comparing the Original and Induced Taxonomies

Table 2 shows much stronger agreement for ASSISTments 2012 than for ASSISTments 2017 at the two reported settings. For 2012, homogeneity, completeness, and normalized mutual information (NMI) are .930, .717, and .809; for 2017, they are .681, .509, and .582. Neither setting yields the highest adjusted Rand index (ARI) or NMI across all tested settings, showing that the settings reported for DKT are not simply those with the closest match. The full comparison results appear in Appendix C; Appendix Table 5 gives additional comparison scores at these two settings. Appendix Figure 5 shows how the 142 questions in noskill are distributed across the induced taxonomy.

Figure 2 shows that questions from large original 2012 categories are often distributed across several induced categories. These divisions may reflect finer mathematical distinctions, recurring wording or templates, or the effect of using more categories. This analysis does not show whether these divisions matter for teaching.

4.2 RQ2: Effect of the Number of Categories

Figure 3 shows how homogeneity and completeness change as K increases. In ASSISTments 2012, homogeneity generally rises while completeness falls: induced categories become more concentrated, while each original KC is distributed across more induced categories. In ASSISTments 2017, both averages generally rise over the tested range, but inverse purity and pairwise recall decline, as shown in Appendix Table 6. The measures therefore capture different effects of using more categories and do not indicate improvement in every aspect of the match.

4.3 RQ3: Taxonomies and DKT Performance

Table 2 reports that the highest test AUC among the induced taxonomies exceeds the AUC with the original taxonomy in both datasets. For ASSISTments 2012, AUC rises from .7367 with the original taxonomy to .7841, a gain of .0474. For ASSISTments 2017, it rises from .7131 to .7403, a gain of .0272. Figure 4 shows that, for `text-embedding-3-large`, test AUC exceeds the original-taxonomy baseline at all 48 PCA/K grid settings in each dataset, including settings with fewer categories than the corresponding original taxonomy within the PB subset.

Before adjustment, many comparison measures correlate strongly with AUC because both vary with PCA dimension and K. After accounting for those patterns and correcting for 15 comparisons, none of the 2012 correlations remains statistically significant. In 2017, several measures of closer

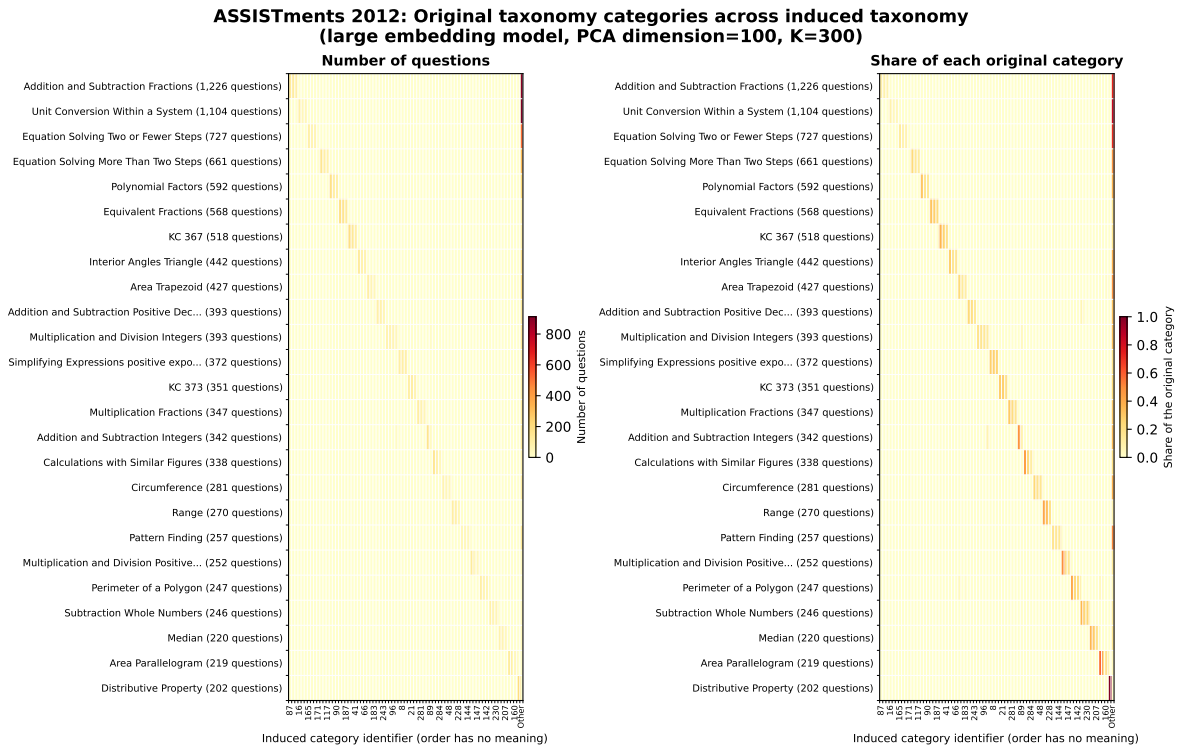


Figure 2: Distribution of questions from the 25 largest categories in the 2012 original taxonomy across the corresponding induced taxonomy. The figure uses `text-embedding-3-large`, PCA$= 100$, and $K = 300$. The left heatmap shows question counts and the right heatmap shows the share of each original category in each induced category. We include the three induced categories containing the most questions from each original category and combine the rest as *Other*. The identifiers for induced categories are arbitrary labels and do not indicate an order.

Dataset	Embedding	PCA	K	Homogeneity	Completeness	NMI	ARI	AUC with original taxonomy	Highest AUC with induced taxonomy	ΔAUC
ASSISTments 2012	3-large	100	300	.930	.717	.809	.293	.7367	.7841	+.0474
ASSISTments 2017	3-large	10	100	.681	.509	.582	.066	.7131	.7403	+.0272

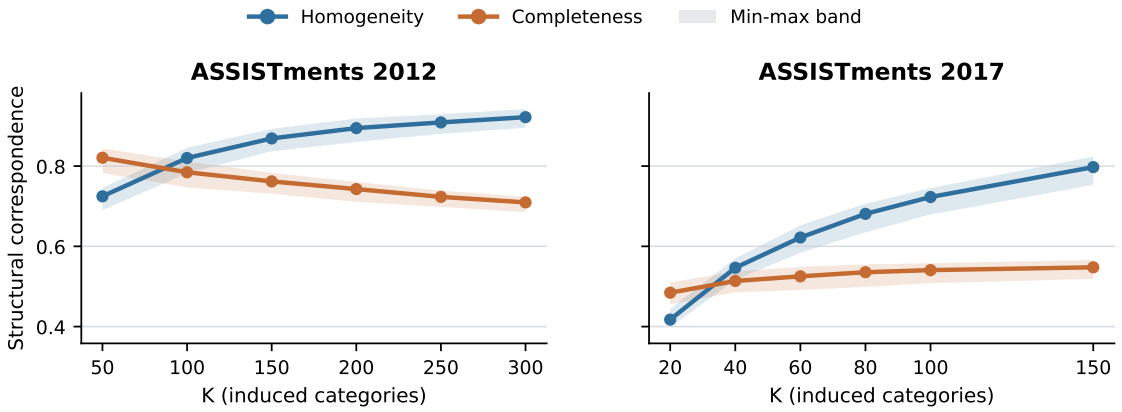
Table 2: Taxonomy comparison and DKT results at the reported settings. Scores for taxonomy agreement are averaged across the two embedding models at each listed PCA dimension and K. The embedding column identifies the model used for the induced taxonomy with the highest observed test AUC. The induced value is the highest observed test AUC across the tested combinations of embedding model, PCA dimension, and K. ΔAUC is induced minus original.

agreement are associated with lower AUC, while a larger variation of information score is associated with higher AUC. Closer agreement is therefore not consistently associated with higher DKT AUC across the tested settings. These correlations do not show that differences from the original taxonomy cause the gains. Appendix D reports the full coefficients in Table 7 and Figure 8.

5 Discussion

The original and induced KC taxonomies organize the same questions using different evidence. The stronger agreement in ASSISTments 2012 suggests that patterns in question text align more closely with the original KC structure there than in ASSISTments 2017. In both datasets, however, original categories are split across several induced categories. These splits may reflect mathematical content, recurring wording, or the use of more induced categories; the current analysis cannot determine which explanation is correct.

The 2017 original taxonomy's largest category, *noskill*, is a non-substantive placeholder covering 142 of 638 questions (22%). Because *noskill* groups questions by the absence of an assigned KC rather than shared content, a taxonomy constructed from question content has no reason to reproduce it as a single category. Appendix Figure 5 shows the induced taxonomy distributing these questions



Bands span eight PCA dimensions and two embedding models (16 configurations per K); they are not confidence intervals.

Figure 3: Mean homogeneity and completeness as the induced taxonomy uses more categories. At each K, the lines show averages over the eight PCA dimensions and two embedding models; bands show the full range across those 16 settings and are not confidence intervals. In 2012, homogeneity generally increases while completeness decreases. In 2017, both generally increase, although inverse purity and pairwise recall decline over the same K range. Appendix Figure 7 gives the full results for each embedding model and PCA dimension.

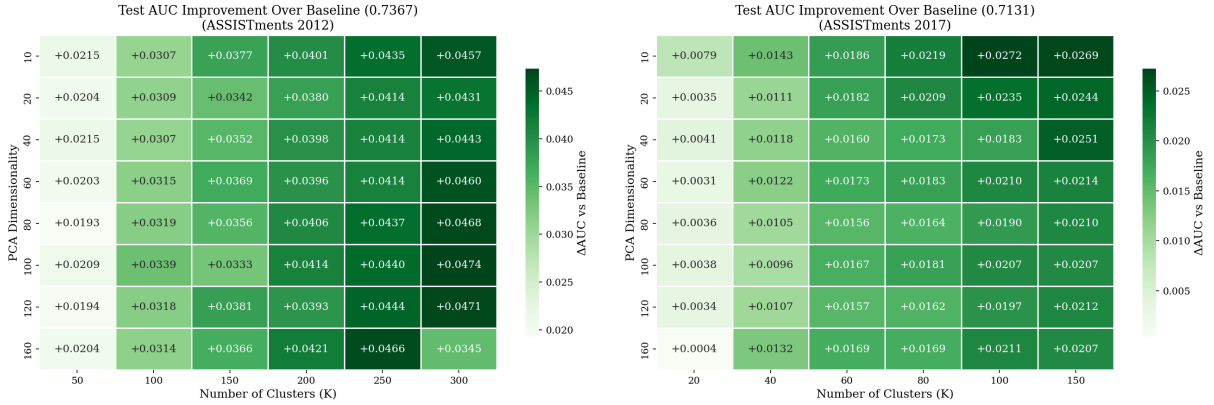


Figure 4: Change in test AUC when using the induced taxonomy instead of the original taxonomy. The figure uses `text-embedding-3-large`, which produced the highest observed test AUC in both datasets. Each cell shows the change in test AUC for one PCA and K setting. The test AUC at every displayed setting exceeds the baseline with the original taxonomy. We also test `text-embedding-3-small`, giving 96 induced taxonomies per dataset in total.

across several categories. The original-taxonomy baseline retains *noskill*, so the reported AUC difference compares DKT with this placeholder against DKT with the complete induced mapping. This redistribution may contribute to both the lower correspondence and the higher AUC observed in 2017, although the experiment does not isolate its contribution from the other changes in the mapping.

The DKT results address a different question: how well each taxonomy supports prediction. The induced taxonomy with the highest observed test AUC improves test AUC by .0474 in 2012 and .0272 in 2017, and Figure 4 shows improvement across all 48 PCA and K settings for `text-embedding-3-large` in both datasets. These results evaluate each induced taxonomy as a whole, including its grouping of questions, number of categories, label frequencies, and model parameters tied to its labels. They do not show which property drives the gain, and AUC measures prediction rather than educational value. After accounting for PCA dimension and K, closer agreement with the original taxonomy is not consistently associated with higher AUC. The negative correlations in 2017 therefore show that lower agreement is sometimes associated with higher AUC, not that disagreement is inherently beneficial.

The induced taxonomies are therefore best treated as complements to systems developed by experts. Their similarities and differences can highlight categories or boundaries for expert review. Determining whether these distinctions matter for

teaching or reflect how students reason requires evidence from educators, student behavior, and solution steps.

A comparison with biological taxonomy helps clarify the idea. Biological categories are sometimes reconsidered using genetic evidence alongside visible traits (San Mauro and Agorreta, 2010). In the same way, text embeddings provide another source of evidence that may support some expert categories and question others. The comparison has a clear limit: similar question text is not automatically a better basis for curriculum design. Educational categories are useful when they support teaching, assessment, or models of learning.

6 Future Work

Future work should extend the analysis to datasets that combine student interactions, question text, and more detailed KC structures. XES3G5M contains 5.5 million responses, assigns multiple KCs to some questions, and links broader skills to narrower ones. DBE-KT22 provides question text and student responses from a university database course, assigns up to four KCs per question, and includes relationships among KCs (Liu et al., 2023; Abdelrahman et al., 2022). These datasets would support induced taxonomies with multiple labels per question or hierarchical relations among categories. The current ASSISTments datasets would remain useful for evaluating models that explicitly support these richer structures. The induced taxonomies could also be evaluated in other tasks that use KCs, including measuring progress with practice, identifying students' relative strengths and weaknesses, and recommending the next practice question (Wang et al., 2020; Cen et al., 2006; Ai et al., 2019).

7 Conclusion

We use embeddings of question text to build educational taxonomies and compare them with the original KC taxonomies in ASSISTments 2012 and 2017. The induced taxonomies preserve much, but not all, of the original structure, with closer agreement in 2012 and consistent changes as the number of induced categories increases. In DKT, the induced taxonomy with the highest observed test AUC improves test AUC by .0474 in 2012 and .0272 in 2017, while closer agreement with the original taxonomy is not consistently associated with higher AUC across the tested settings. The central result therefore separates two questions: similarity to a taxonomy developed by experts and usefulness for prediction. Taxonomies built from embeddings offer a promising alternative for student models, but their educational value still requires review by experts and evidence from students.

8 Limitations

Our analysis covers only ASSISTments 2012 and 2017 questions with linked text. Both datasets come from the same learning platform, so the findings may not extend to other platforms, courses, schools, languages, or educational settings. We include only responses tied to questions with text.

The procedure for building the taxonomy assigns each question to one induced category. It does not allow several KCs per question, organize KCs from broad to narrow, or model student reasoning. The induced taxonomy may organize questions by wording, templates, formatting, answer type, or source rather than by the mathematics or knowledge required to answer them.

The comparison measures show how closely the two taxonomies group the same questions rather than their educational value. Purity and homogeneity can rise simply with the number of categories, and overall scores do not identify which specific splits or mergers matter for instruction. We do not analyze solution steps, inspect individual category boundaries, or ask educators to judge the categories, so we cannot determine the educational value of those differences.

The experiments use two models for embedding text, one clustering method, one clustering seed, and a fixed set of PCA dimensions and values of K. The observed results may therefore depend on the tested models, clustering procedure, category count, question templates, or distribution of cluster sizes. We use PCA as the only dimensionality-reduction method; alternative reduction methods may produce different induced taxonomies. We do not directly benchmark against prior KC-discovery methods such as SMART or KCluster.

We evaluate one DKT architecture. Although all taxonomies use the same records, student split, training procedure, search protocol, and trial budget, we tune DKT separately for each taxonomy, so the selected hyperparameters may differ. Changing K also changes the number of KC labels and can change the number of model parameters tied to those labels. Without random-label baselines

matched on the number and sizes of categories, the experiment cannot separate grouping by meaning from the effects of category count, how often each label appears, or model size.

The reported induced AUC is the highest result across all tested combinations of embedding model, PCA dimension, and K. Because this combination is identified using test AUC, the reported gain summarizes the best tested result rather than a combination chosen before looking at test performance. Figure 4 shows the results across the full grid. For each combination, the Bayesian search chooses DKT settings using validation AUC and does not choose the embedding model, PCA dimension, or K. Because students rather than questions are held out, test AUC measures performance for new students on questions used to build the taxonomy, not entirely new questions. Larger K may also make induced labels behave more like coarse question identifiers, which could contribute to higher AUC. Finally, the analysis of taxonomy agreement and AUC shows an association and does not establish that changing agreement causes the AUC gains.

Acknowledgments

We thank the ASSISTments Project at Worcester Polytechnic Institute for providing the Problem Bodies data, which allowed us to link records of student responses with question text (ASSISTments Project, n.d.).

References

Ghodai Abdelrahman, Sherif Abdelfattah, Qing Wang, and Yu Lin. 2022. DBE-KT22: A knowledge tracing dataset based on online student evaluation. *arXiv preprint arXiv:2208.12651*.

Ghodai Abdelrahman, Qing Wang, and Bernardo Nunes. 2023. Knowledge tracing: A survey. *ACM Computing Surveys*, 55(11):224:1–224:37.

Fangzhe Ai, Yishuai Chen, Yuchun Guo, Yongxiang Zhao, Zhenzhu Wang, Guowei Fu, and Guangyan Wang. 2019. Concept-aware deep knowledge tracing and exercise recommendation in an online learning system. In *Proceedings of the 12th International Conference on Educational Data Mining*, pages 240–245. International Educational Data Mining Society.

ASSISTments Project. ASSISTments Problems [online]. n.d. Accessed August 1, 2026.

Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300.

Hao Cen, Kenneth Koedinger, and Brian Junker. 2006. Learning factors analysis – a general method for cognitive model evaluation and improvement. In *Intelligent Tutoring Systems*, pages 164–175, Berlin, Heidelberg. Springer.

Common Core State Standards Initiative. Development process [online]. n.d. Accessed July 31, 2026.

Albert T. Corbett and John R. Anderson. 1995. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278.

Mingyu Feng, Neil Heffernan, and Kenneth Koedinger. 2009. Addressing the assessment challenge with an online system that tutors as it assesses. *User Modeling and User-Adapted Interaction*, 19(3):243–266.

Edward B. Fowlkes and Colin L. Mallows. 1983. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569.

Kenneth R. Koedinger, Albert T. Corbett, and Charles Perfetti. 2012. The knowledge–learning–instruction framework: Bridging the science–practice chasm to enhance robust student learning. *Cognitive Science*, 36:757–798.

Qi Liu, Zhenya Huang, Yu Yin, Enhong Chen, Hui Xiong, Yu Su, and Guoping Hu. 2021. EKT: Exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering*, 33(1):100–115.

Zitao Liu, Qiongqiong Liu, Jiahao Chen, Shuyan Huang, Jiliang Tang, and Weiqi Luo. 2022. pyKT: A Python library to benchmark deep learning based knowledge tracing models. In *Advances in Neural Information Processing Systems*, volume 35, pages 18542–18555. Curran Associates, Inc.

Zitao Liu, Qiongqiong Liu, Teng Guo, Jiahao Chen, Shuyan Huang, Xiangyu Zhao, Jiliang Tang, Weiqi Luo, and Jian Weng. 2023. XES3G5M: A knowledge tracing benchmark dataset with auxiliary information. In *Advances in Neural Information Processing Systems*, volume 36, pages 32958–32970.

Noboru Matsuda, Jesse Wood, Raj Shrivastava, Machi Shimmei, and Norman Bier. 2022. Latent skill mining and labeling from courseware content. *Journal of Educational Data Mining*, 14(2):1–31.

Marina Meilă. 2007. Comparing clusterings, an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895.

Hyeongdon Moon, Richard Lee Davis, Seyed Parsa Neshaei, and Pierre Dillenbourg. 2025. Using large multimodal models to extract knowledge components

for knowledge tracing from multimedia question information. In *Proceedings of the 18th International Conference on Educational Data Mining*, pages 342–353. International Educational Data Mining Society.

OpenAI. 2026. Vector embeddings. Accessed 2026-08-07.

Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, volume 28, pages 505–513.

Andrew Porter, Jennifer McMaken, JeongSuk Hwang, and Rui Yang. 2011. Common core standards: The new U.S. intended curriculum. *Educational Researcher*, 40(3):103–116.

Ethan Prihar, Manaal Syed, Korinn Ostrow, Stacy Shaw, Adam Sales, and Neil Heffernan. 2022. Exploring common trends in online educational experiments.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3982–3992.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 410–420. Association for Computational Linguistics.

Diego San Mauro and Ana Agorreta. 2010. Molecular systematics: A synthesis of the common methods and the state of knowledge. *Cellular and Molecular Biology Letters*, 15(2):311–341.

Yu Su, Qingwen Liu, Qi Liu, Zhenya Huang, Yu Yin, Enhong Chen, Chris Ding, Si Wei, and Guoping Hu. 2018. Exercise-enhanced sequential modeling for student performance prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1):2435–2443.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(95):2837–2854.

Fei Wang, Qi Liu, Enhong Chen, Zhenya Huang, Yuying Chen, Yu Yin, Zai Huang, and Shijin Wang. 2020. Neural cognitive diagnosis for intelligent education systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6153–6161.

Yumou Wei, Paulo Carvalho, and John Stamper. 2025. KCluster: An LLM-based clustering approach to knowledge component discovery. In *Proceedings of the 18th International Conference on Educational Data Mining*, pages 228–240. International Educational Data Mining Society.

Xiaolu Xiong, Siyuan Zhao, Eric G. Van Inwegen, and Joseph E. Beck. 2016. Going deeper with deep knowledge tracing. In *Proceedings of the 9th International Conference on Educational Data Mining*, pages 545–550. International Educational Data Mining Society.

A Question Text Inputs and Examples

Question examples. The embedding models use only the cleaned text from the `problem_body` field in `questions.csv`; grade information and the original category labels are excluded. The grade labels range from elementary school through high school mathematics and do not necessarily match a student's enrolled grade. Although Grades 6–8 usually correspond to U.S. middle school, the data cover a broader range. The Grade 4 and Grade 6 examples also illustrate that converting web pages to plain text can remove mathematical layout and diagrams. Images are not provided to the models. Because the access terms for Problem Bodies prohibit quoting the questions, we describe representative examples instead (ASSISTments Project, n.d.).

- **ASSISTments 2012, Grade 4:** converting an improper fraction.
- **ASSISTments 2012, Grade 6:** finding a square's area from a diagram.
- **ASSISTments 2017, Grade 7:** finding a percentage of a whole.
- **ASSISTments 2017, Grade 8:** using the formula for a cone's volume.
- **ASSISTments 2012, High School Algebra:** comparing the degrees of polynomial terms.

B Taxonomy and DKT Settings

By default, OpenAI's `text-embedding-3-small` and `text-embedding-3-large` produce embeddings with 1,536 and 3,072 dimensions, respectively. We use the API's `dimensions` parameter to request outputs with 768 and 1,536 dimensions, respectively (OpenAI, 2026). Let e_i denote the embedding for question i. PCA reduces it to $z_i = P(e_i)$, and k-means assigns z_i to the nearest centroid:

$$c_i = \arg \min_{j \in \{1,\dots,K\}} \|z_i - \mu_j\|_2^2.$$

Here, c_i is the label assigned by the induced taxonomy, K is the number of induced categories, and μ_j is the centroid of cluster j. We test PCA dimensions $\{10, 20, 40, 60, 80, 100, 120, 160\}$. We test values of K $\{50, 100, 150, 200, 250, 300\}$ for 2012 and $\{20, 40, 60, 80, 100, 150\}$ for 2017. PCA and k-means use a random seed of 42. For each setting, k-means uses 10 initializations and at most 300 iterations (`n_init=10`, `max_iter=300`).

For response t, q_t denotes the question, s_t its original KC label, r_t whether the response is correct, and $c(q_t)$ the label assigned by the induced taxonomy. We apply the relabeling

$$(q_t, s_t, r_t) \mapsto (q_t, c(q_t), r_t).$$

For 2012, we write induced labels to `skill_id`. For 2017, we write them as text in `skill`. pyKT then renumbers them as consecutive concept IDs.

For the baseline that uses the original taxonomy and for every induced taxonomy, we run a separate Weights & Biases Bayesian search that maximizes validation AUC. Each induced taxonomy is defined by its dataset, embedding model, PCA dimension, and K. Within each dataset, all searches use the same candidate settings, number of trials, assignment of students to training, validation, and testing, training procedure, and selection rule.

For 2012, the search varies `emb_size` over $\{64, 128, 256, 512\}$, `learning_rate` over $\{0.01, 0.001, 0.0007, 0.0005\}$, `dropout` over $\{0.1, 0.2, 0.3, 0.5\}$, and `seed` over $\{42, 84\}$. For 2017, the search varies `emb_size` over $\{64, 96, 128, 192, 256, 384, 512, 768, 1024\}$, `learning_rate` over $\{0.01, 0.001, 0.0007, 0.0005, 0.002, 0.0002\}$, and `dropout` over $\{0.01, 0.1, 0.2, 0.3, 0.4, 0.45, 0.5\}$. The 2017 search fixes `seed` at 42. For 2012, seed is one of the settings considered by the search; it is not used to repeat each taxonomy.

Within each search, we select the completed run with the highest validation AUC and report its test AUC; the embedding model, PCA dimension, and K remain fixed outside the search.

Both the DKT/QID input format, which includes question and KC identifiers, and fold 0 are fixed across all searches. Training uses Adam with batch size 256, at most 200 epochs, and a maximum sequence length of 200. The preprocessing code reserves 20% of students for testing with `random_state=1024`; the remaining students are divided into five folds, yielding the approximate 64/16/20 training/validation/test split. Training stops after 30 epochs without a validation AUC improvement of at least .001.

C Measures for Comparing the Taxonomies

Combining eight PCA dimensions with six values of K gives 48 settings per dataset. We evaluate both embedding models at every setting, yielding 96 rows per dataset. Each analysis below states whether results for the two embedding models are reported separately or averaged.

At the two reported combinations of PCA dimension and K, the average spread in cluster size across the two embedding models is .591 for 2012 and .447 for 2017. This spread is the standard deviation divided by the mean; a larger value means the

Measure	What the score measures
ARI	Adjusted Rand index: agreement between pairs of questions, adjusted for chance; higher is closer.
NMI / AMI	Normalized / adjusted mutual information: how much information the two taxonomies share; AMI adjusts for chance; higher is closer.
Homogeneity	Whether each induced category mostly contains questions from one original category; higher is closer.
Completeness	Whether questions from each original category mostly stay together in one induced category; higher is closer.
V-measure	Harmonic mean of homogeneity and completeness; higher is closer.
Macro purity	Average share with the most common original label in each induced category, weighting induced categories equally; higher is closer.
Global purity	Share of all questions with the most common original label in their induced category, giving larger induced categories more weight; higher is closer.
Global / macro inverse purity	How much of each original category remains in its largest induced category, weighting categories by size or equally; higher is closer.
FM	Fowlkes–Mallows: a combined score based on pairwise precision and recall; higher is closer.
Pairwise P/R/F_1	Precision for pairs in the same induced category, recall for pairs with the same original label, and their F_1; higher is closer.
VI	Variation of information: how different the two taxonomies are based on their information content; lower is closer.

Table 3: Measures used to compare the original and induced taxonomies (Rosenberg and Hirschberg, 2007; Vinh et al., 2010; Meilă, 2007; Fowlkes and Mallows, 1983). Increasing K can raise some scores and lower others, so the scores should be interpreted together.

Dataset	Best ARI	Best AMI	Best V/NMI	Best FM	Lowest VI
2012	.490 (120, 50)	.812 (160, 150)	.830 (160, 150)	.503 (120, 50)	1.554 (160, 150)
2017	.114 (100, 40)	.371 (100, 80)	.663 (120, 150)	.165 (80, 20)	2.789 (120, 150)

Table 4: Best agreement scores across the 48 settings, averaged across the two embedding models. ARI is the adjusted Rand index; AMI and NMI denote adjusted and normalized mutual information, respectively; FM is Fowlkes–Mallows; and VI is variation of information. Parentheses give the PCA dimension and K. Lower VI means a closer match; higher values mean a closer match for the other measures. NMI and V-measure have the same values here and reach their highest values at the same setting. Table 6 shows these and the purity measures at the smallest and largest values of K. Figure 6 shows ARI and NMI separately for each embedding model and dataset.

Dataset (PCA, K)	Homog.	Complete.	V/NMI	Global pur.	Macro pur.	Global inv.	Macro inv.	ARI	AMI	FM	Pair F_1	VI
2012 $(100, 300)$	.930	.717	.809	.888	.898	.406	.617	.293	.776	.393	.299	1.872
2017 $(10, 100)$	.681	.509	.582	.530	.546	.266	.486	.066	.272	.123	.083	3.290

Table 5: Structural scores at the two PCA/K settings used for the reported DKT results. Each score is averaged across the two embedding models. "Homog." means homogeneity, "Complete." means completeness, "pur." means purity, and "inv." means inverse purity. Table 3 defines ARI, AMI, and NMI; FM means Fowlkes–Mallows. VI uses natural logarithms and is the only score for which lower values mean a closer match. V-measure and NMI have the same values here.

cluster sizes vary more. The average share of questions in the largest cluster is 1.28% for 2012 and 2.51% for 2017. Neither 2012 embedding model produces a cluster with only one question; each 2017 embedding model produces one such cluster. No model places more than 3% of questions in its largest cluster. All K clusters contain at least one question. The mean cluster size is 62.45 for 2012 and 6.38 for 2017.

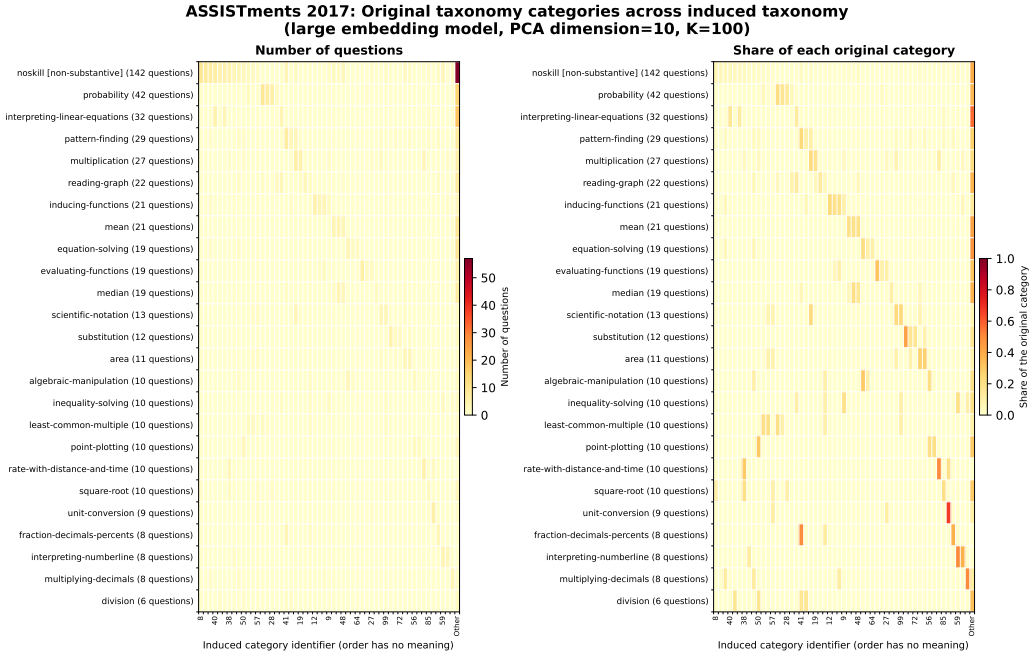
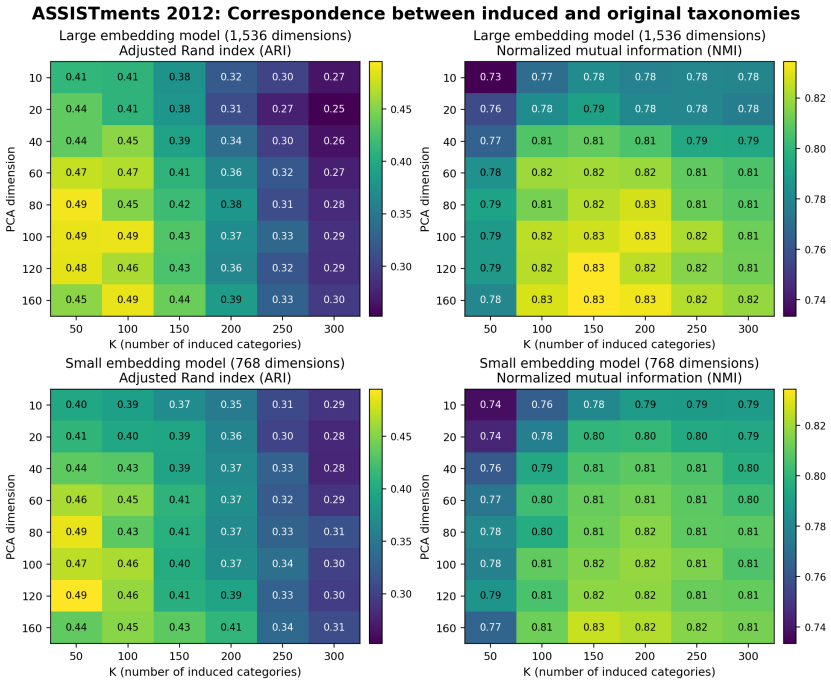


Figure 5: Distribution of questions from the 25 largest categories in the 2017 original taxonomy across the corresponding induced taxonomy. The figure uses `text-embedding-3-large`, PCA$= 10$, and $K = 100$. The left heatmap shows question counts and the right heatmap shows the share of each original category in each induced category. We include the three induced categories containing the most questions from each original category and combine the rest as *Other*. Induced category identifiers are arbitrary labels and do not indicate an order.

Dataset/K	Homog.	Complete.	Global purity	Global inverse	Pair precision	Pair recall	VI	Size spread
2012/50	.725	.821	.572	.717	.426	.516	1.854	.522
2012/300	.922	.710	.880	.392	.839	.176	1.948	.575
2017/20	.417	.484	.345	.439	.175	.146	3.460	.418
2017/150	.797	.548	.674	.270	.442	.042	2.901	.476

Table 6: Scores at the smallest and largest tested values of K. Each value is the simple average over eight PCA dimensions and both embedding models. In both datasets, increasing K raises homogeneity, global purity, and pairwise precision. It lowers global inverse purity and pairwise recall. In 2012, completeness falls and VI rises. In 2017, completeness rises and VI falls. "Homog." means homogeneity, "Complete." means completeness, and VI means variation of information. Size spread is the standard deviation of the induced cluster sizes divided by their mean; a larger value means that the cluster sizes differ more.

(a) ASSISTments 2012



(b) ASSISTments 2017

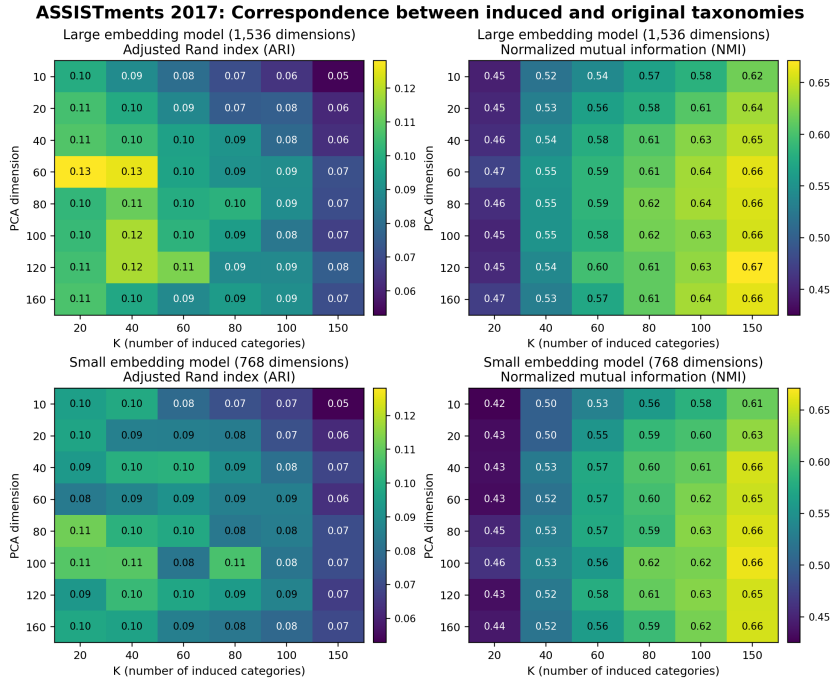
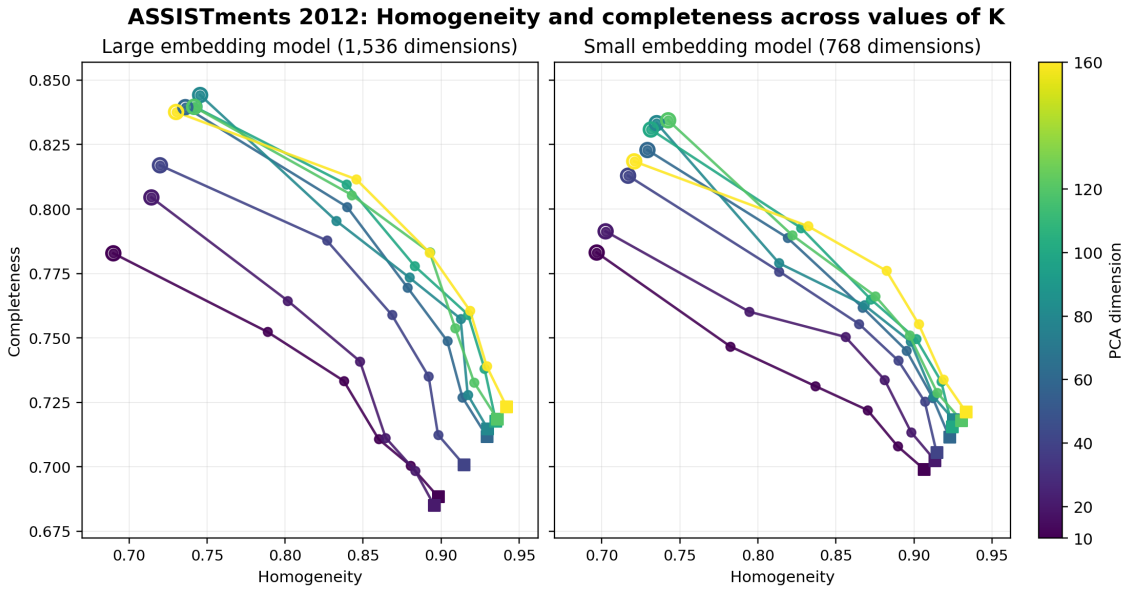


Figure 6: ARI and NMI for every PCA dimension and value of K, shown separately for each embedding model in 2012 (a) and 2017 (b). Table 4 gives the highest values after averaging scores across the two models. In 2017, ARI and NMI reach their highest values at different settings, and, for each metric, each embedding model's highest value differs from the highest value obtained after averaging scores across the two models.

(a) ASSISTments 2012



(b) ASSISTments 2017

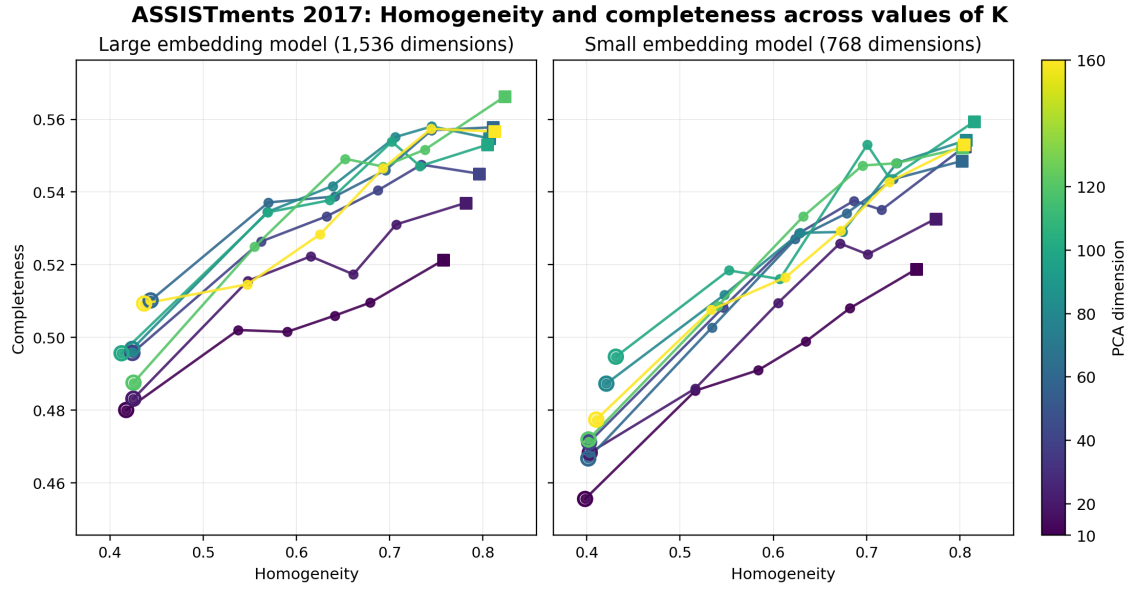


Figure 7: Homogeneity and completeness across values of K, shown separately for each combination of embedding model and PCA dimension in 2012 (a) and 2017 (b). In 2012, homogeneity generally rises while completeness falls. In 2017, both generally rise across the tested settings, although global inverse purity and pairwise recall fall from the smallest to the largest tested value of K, as shown in Table 6. Outlined circles mark the smallest tested K, and squares mark the largest.

D Relation Between Taxonomy Agreement and AUC

To examine the relationship between taxonomy agreement and AUC, we analyze 48 matched settings of PCA dimension and K per dataset. At each setting, we average each structural score across the two embedding models and separately average the two embedding-specific test AUC values. The 2017 AUC results include three additional settings, $(10, 200)$, $(10, 250)$, and $(10, 300)$; we exclude them because the taxonomy comparison has no matching scores.

We use Spearman correlation to test whether closer taxonomy agreement is associated with higher test AUC. We then repeat the analysis after ranking both measures and removing patterns linked to PCA dimension and K. Because we test 15 measures, we apply the Benjamini–Hochberg correction separately by dataset and separately before and after this adjustment.

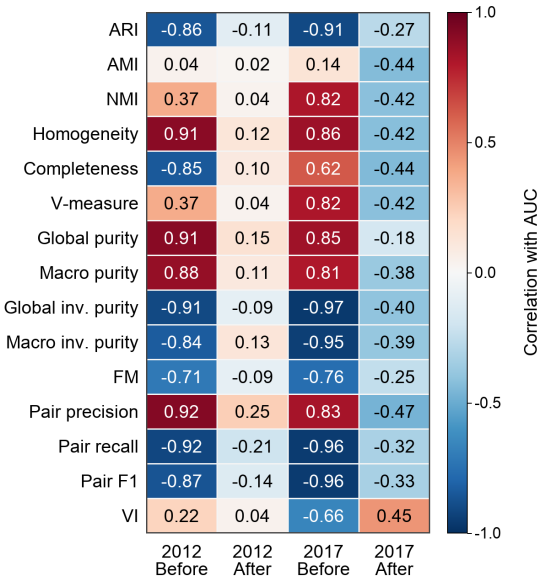


Figure 8: Rank correlations between each taxonomy-agreement score and test AUC, both averaged across the two embedding models. "After" indicates that rank patterns linked to PCA dimension and K have been removed. Cells show coefficients; Table 7 reports significance after correction.

Measure	2012 before	2012 after	2017 before	2017 after
ARI	$-.858^{*}$	$-.109$	$-.908^{*}$	$-.273$
AMI	$.041$	$.024$	$.138$	$-.436^{*}$
NMI	$.374^{*}$	$.038$	$.816^{*}$	$-.418^{*}$
Homogeneity	$.906^{*}$	$.125$	$.857^{*}$	$-.425^{*}$
Completeness	$-.847^{*}$	$.097$	$.620^{*}$	$-.443^{*}$
V-measure	$.374^{*}$	$.038$	$.816^{*}$	$-.418^{*}$
Global purity	$.913^{*}$	$.150$	$.851^{*}$	$-.179$
Macro purity	$.879^{*}$	$.105$	$.810^{*}$	$-.376^{*}$
Global inverse purity	$-.908^{*}$	$-.094$	$-.968^{*}$	$-.403^{*}$
Macro inverse purity	$-.836^{*}$	$.126$	$-.946^{*}$	$-.387^{*}$
Fowlkes–Mallows	$-.714^{*}$	$-.088$	$-.763^{*}$	$-.246$
Pairwise precision	$.922^{*}$	$.246$	$.833^{*}$	$-.470^{*}$
Pairwise recall	$-.917^{*}$	$-.207$	$-.959^{*}$	$-.324$
Pairwise F_1	$-.866^{*}$	$-.139$	$-.960^{*}$	$-.330$
VI	$.224$	$.040$	$-.664^{*}$	$.446^{*}$

Table 7: Associations, across the 48 shared settings of PCA dimension and K in each dataset, between each taxonomy-agreement score and test AUC, both averaged across the two embedding models. "Before" is the direct Spearman correlation; "After" is the correlation after removing rank patterns linked to PCA dimension and K. An asterisk marks $p < .05$ after the Benjamini–Hochberg correction, applied separately by dataset and analysis stage. VI is a distance measure, so a positive coefficient indicates that a greater difference between the taxonomies is associated with higher test AUC.

Numbers Through Signs: How LLMs Connect Counts, Number Words, and Numerals

J. Albert Yoo
Eötvös Loránd University
j.albert.yoo.academic@gmail.com

Abstract

Large language models manipulate numerical expressions fluently, yet it is unclear whether a number inferred from a collection enters the same internal interface as an explicit numeral or number word. We test digits, English number words, and counts of repeated tokens in five Qwen3 base models from 0.6B to 14B. After each quantity expression but before a comparison, parity, or equality question, we interchange one residual-stream state between forms. Textual-count states preserve held-out cardinal order, even at fixed sequence length. In behaviourally competent 8B and 14B cases, digit and number-word states are bidirectionally interchangeable. At the tested collection boundary, however, a count state does not transfer even to a different count of the same form, and no count–sign direction meets the same one-shot criteria. A same-depth sweep and symbol-derived linear subspace do not recover a bidirectional local interface. Longer interventions and a separate mixture-of-experts model reveal strong but directional bridges. Counting information and correct behaviour can therefore coexist without a locally portable state at the tested boundary.

1 Introduction

Large language models (LLMs) can manipulate equations, check proof steps, and answer many mathematical questions, while remaining unexpectedly brittle at counting repeated items or extracting quantities from ordinary text. Recent mechanistic work complicates any simple explanation based on missing number information. Counts can be decoded from language- and vision-language-model activations, and multi-stage routines can transmit and aggregate counting signals (Hasani et al., 2026b,a). A model may therefore contain a count even when its answer is unreliable. The open issue is how that count participates in the rest of the model's numerical computation.

Related studies find regular geometry across explicit number forms (Štefánik et al., 2026), and show that numerical information recoverable by a probe can exceed what a model expresses in its answers (Yuchi et al., 2026). Low-dimensional, format-agnostic subspaces can also transfer reasoning more effectively than replacing an entire hidden state (Yuan and Su, 2026). These results establish numerical structure and usable signals. They leave a sharper question unanswered: once the model has constructed the cardinality of a collection, can that state be used in place of the state produced by a digit or number word in a later operation?

The question matters because human numerical expressions are ordinarily reused across very different encounters. The mark *3*, the word *three*, three objects, and three repeated actions can all guide a new comparison, match, or calculation. Humans develop these abilities through language embedded in perception, action, and cultural practice. A text-first LLM follows a strikingly different route: proofs, numerals, descriptions of objects, and reports of action all reach it through the same stream of signs. This makes the model an inspectable case for separating fluency with public mathematical notation from the internal connection between signs and constructed quantities.

We study that connection with a task-late interchange assay. A Qwen3 model first reads a quantity as a digit, number word, or enumerated textual collection. Only after the expression ends does it receive a comparison, parity, or equality question. We transplant the boundary state from one form into another and ask whether the model's answer follows the donor number. The experiments show a robust digit–word interface, but the tested collection-boundary state is not portable even between two textual counts. Partial transfer under longer interventions shows that the count–sign connection is graded rather than absent. For mathematical NLP, the result separates what behavioural ac-

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 285–290
October 29, 2026 ©2026 Association for Computational Linguistics

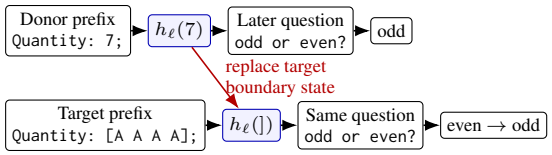


Figure 1: **One-shot task-late interchange.** The donor encodes seven as a digit; the target encodes four as a textual count. Both prefixes end before the parity question. If the transplanted state can play the donor's numerical role, the target answer changes from even to odd. A same-number donor tests whether cross-form replacement preserves the original answer.

curacy and numerical probes measure from whether different expressions feed a common, immediately reusable numerical state.

2 Task-Late Interchange Assay

Models, number forms, and tasks. We study base Qwen3 checkpoints at 0.6B, 1.7B, 4B, 8B, and 14B parameters (Yang et al., 2025). Production runs use pinned revisions, BF16 weights and activations, and one A100 80GB GPU, without quantization or CPU offload. A Qwen3-30B-A3B checkpoint is analysed separately because sparse expert routing makes it an architectural contrast rather than a sixth dense scale point. Cardinalities 2–9 are expressed as an Arabic digit (`3`), an English number word (`three`), a variable-length textual count (`[A A A]`), or a shuffled ten-item A/B sequence containing three A tokens. The last condition keeps total length constant while cardinality changes. The later questions ask whether the quantity exceeds a threshold, is odd or even, or equals a threshold.

Interchange by example. Figure 1 shows the central intervention. The donor prefix states seven explicitly; the target prefix presents four repeated tokens. At one decoder layer, we replace the residual-stream state at the target expression's closing bracket with the donor state at the digit. The unmodified target answers *even*; successful counterfactual transfer makes the continuation follow seven and answer *odd*. Replacing the boundary state with one derived from a different form of the same number should preserve the target's answer. Together, counterfactual transfer and same-number preservation test *local causal interoperability*: whether one task-late state can assume another form's numerical role in the model's own downstream computation (Geiger et al., 2025).

Selection, screening, and measures. For each model, the decoder site is selected using development-template digit–word transfer only; the embedding output is excluded. The selected sites for the dense series are 3, 4, 6, 6, and 1 in ascending model size. Evaluation wording is chosen through behaviour alone: digit and word balanced accuracy must each reach .85, textual-count accuracy .65, each count answer class must exceed .60, and every principal direction must retain at least 16 trials on which the unmodified donor and target are correct. Conditional interchange intervention accuracy (IIA) is the proportion of those eligible trials whose patched answer follows the donor. A same-number donor supplies a do-no-harm preservation test; exact no-op and wrong-donor trials check the intervention itself. A direction is counted as established when eligibility is at least .50, same-number preservation .90, one-shot IIA .80, and the lower cluster-bootstrap bound on donor-directed logit gain is positive.

Persistence, readouts, and local scope. The primary intervention is applied once; additional windows maintain the donor state over 2, 4, 8, or all remaining decoder sites. For 14B comparison, we also begin a to-end intervention at every same-depth site. Ridge readouts leave out one numerical value at a time, and a rank-7 cardinal subspace is learned only from digit and word development states. The assay targets one local interface, not an information bottleneck: after the closing-bracket state is replaced, later tokens can still attend to the original A tokens.

3 Results

Collection states preserve cardinal order. A ridge readout trained on textual-count states orders unseen values with Spearman $\rho = .990$–$.993$ across all five dense models (Fig. 2a). The fixed-total A/B control reaches $\rho = .967$–$.993$, so total sequence length is unnecessary for this within-format result. A digit-trained readout transfers to number words ($\rho = .891$–$.956$). Applied to textual counts, however, the same readout gives $\rho = -.707$ to $-.967$. The observed rank reversal is directional: digit-trained readouts also give $\rho = -.775$ to $-.922$ on fixed-total collections. Varying sequence length therefore does not explain this systematic misalignment of the tested linear coordinates, whose origin remains unresolved.

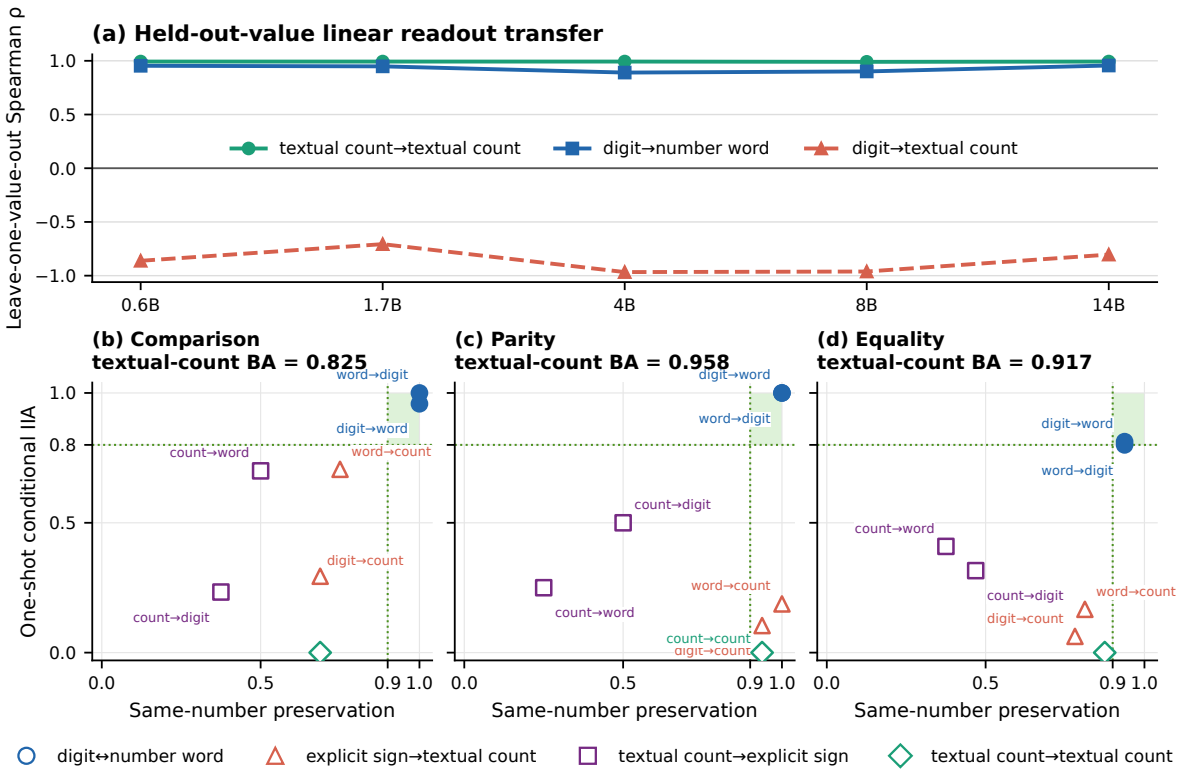


Figure 2: **Cardinal readouts and one-shot interchange.** (a) A readout trained on textual counts preserves held-out numerical order; a digit-trained readout transfers to number words but not to textual counts. The fixed-total control reaches $\rho = .967$–$.993$ within format. (b–d) Same-number preservation and conditional IIA for Qwen3-14B on the three behaviourally screened tasks, including the previously unreported textual-count-to-textual-count baseline. Filled points satisfy the four stated one-shot criteria.

Digits and number words provide the causal anchor. Qwen3-14B passes behavioural screening on textual counts for comparison (.825 balanced accuracy), parity (.958), and equality (.917). Across all three operations, digit and number-word states meet the one-shot criteria in both directions. Comparison and parity have same-number preservation of 1.000 and IIA of at least .958 in each direction; equality has preservation of .938 and IIA of .800–.813. The positive anchor demonstrates that the assay can detect a locally portable state shared by two expression forms.

Competent counts do not show the same one-shot interface. Parity supplies the clearest contrast because every requested 14B count–sign intervention has correct unmodified endpoints: each direction contains 48 eligible trials, versus 24 for each explicit-sign direction. Digit-to-count replacement preserves a same-number answer at .938 but follows a different donor number on only .104 of trials; number-word-to-count preservation is 1.000 while IIA is .188. The two explicit-sign directions score 1.000 on both measures. The 8B model independently passes an equality wording with textual-count balanced accuracy .813: both digit–word directions meet the criteria, while none of the four count–sign directions does. Across comparison, parity, and equality, zero of twelve 14B count–sign directions establish the one-shot local interface.

The count boundary is not a portable register. The full format matrix supplies a same-form baseline. At the selected 14B site, count-to-count one-shot IIA is zero for comparison, parity, and equality. Parity is decisive: all 48 trials are eligible, and preservation is .938 over 16 same-number trials. Thus the tested closing-bracket state does not carry a counterfactually portable count even within the collection format; the cross-form null is not merely disagreement between two portable codes.

Depth and linear-subspace controls preserve the dissociation. For 14B comparison, to-end replacement beginning at every same-depth site finds digit–word transfer at many starting depths, but no identity-valid bidirectional count–sign relation. A rank-7 subspace learned from digit and

number-word states gives IIA and preservation of 1.000 within the explicit sign pair. For count-to-digit, subspace replacement raises preservation to 1.000 while IIA remains .067. No count direction becomes bidirectional, nor do equal-rank random, label-permuted, or top-variance controls reveal one.

Longer interventions reveal partial bridges. Count–sign replacements frequently move the output margin towards the donor even when the categorical answer does not change. Maintaining the transplanted state can produce much stronger transfer: on 14B parity, number-word-to-count IIA rises from .188 after one site to .729 over eight. The separate 30B-A3B mixture-of-experts model provides a stronger boundary case on one equality wording. Digit-to-count reaches .806 at window 8, and number-word-to-count reaches .833 at windows 2–8, all with same-number preservation 1.000. The reverse directions and a second equality wording are weaker. Text-only training can therefore connect explicit signs to a constructed count, but the strongest connections remain directional and dependent on prompt or continued intervention.

4 Discussion

The local failures have different immediate diagnostics. Count-to-count and sign-to-count results show that the closing bracket is not a causal bottleneck: the model can continue to use the surviving collection tokens, while sustained sign-to-count intervention increasingly competes with that evidence. Full-state count-to-sign replacement can also disrupt an incompatible sign slot. Yet the symbol-subspace count-to-digit intervention preserves the same-number answer perfectly while transferring the donor on only .067 of eligible trials. Behavioural numeracy, probe decodability, causal influence, and immediate cross-format reuse are therefore distinct measurements for mathematical NLP.

The experiments separate properties often collapsed into saying that a model "represents" a number: recoverable information, behavioural use, causal influence, and immediate reuse across expressions. Broader coordination could instead be distributed across states and remain flexible across directions, wordings, and operations. The tested Qwen3 states separate information and influence from this local form of coordination.

For humans, learning that *3* means *three* is more than memorizing a pair of marks: both signs can be used for a quantity already encountered in counting, matching, and acting on collections. Qwen3's digit–word interchange shows that language training can coordinate signs internally. The symbol-grounding problem asks how such signs connect beyond one another (Harnad, 1990); model tokens may also inherit derivative reference from the human practices recorded in their data (Mandelkern and Linzen, 2024). Our dissociation concerns how far that inherited practice organises the model's own collection-derived states.

Embodied accounts suggest why human numerical uses may integrate readily (Lakoff and Núñez, 2000; Winter and Yoshimi, 2020). The body need not store an abstract object called three. It can provide a common causal ground: matching, repetition, distribution, addition, removal, and persistence constrain signs and situations through recurring successes and failures. The familiar *felt threeness* of three objects or actions marks a phenomenological contrast; our experiment tests functional organisation, not experience. Infants distinguish small-set tracking from approximate magnitude systems (Feigenson et al., 2004), while non-human animals act selectively on quantity with limited public symbolism (Nieder, 2021). Text-first LLMs expose the complementary case: extensive public symbolism without a history of quantity-guided action. How nonsymbolic quantity connects to exact number words remains disputed (Leibovich and Ansari, 2016).

The positive results sharpen this contrast. Qwen3-14B coordinates explicit signs tightly, while the MoE model develops strong symbol-to-count transfer under sustained intervention. Such directional, unstable bridges differ from flexible reuse across human numerical practice. Testing embodied development requires changing how numerical forms are learned, not further interpreting one pretrained text-only family.

5 Conclusion

Textual counts carry cardinal information and enter partial causal bridges, but do not share the portable digit–word interface. At the tested boundary in competent 8B and 14B models, a count state transfers neither within its format nor to an explicit sign after one shot. Same-depth and symbol-subspace tests do not restore bidirectional transfer; longer interventions reveal a fragile connection.

Limitations

The five dense checkpoints belong to one model family and do not provide a controlled scaling law. Cardinalities are restricted to 2–9, and every collection is textual. Three comparison, three parity, and four equality wordings were screened; one 14B wording per operation passed the behaviour-only rule. The principal causal results therefore concern prompts with adequate baseline competence rather than prompt variation in general. The cluster-bootstrap intervals summarise variation over eight source values within this finite synthetic design; they do not estimate a population over numbers, prompts, model families, or independently trained checkpoints. The 30B-A3B contrast changes sparse routing, total parameter count, active parameter count, and other properties together.

The intervention replaces one task-late token. The original items in a textual collection remain available through attention, allowing a counterfactual donor state to compete with surviving target evidence. Source and target states are compared only at the same decoder depth. The experiments therefore leave open distributed, multi-token, nonlinear, temporally shifted, and collection-trained interfaces. The all-depth sweep covers same-depth correspondences, while ridge readouts and the rank-7 symbol-derived subspace test linear coordinates learned from specific development templates.

The operations and thresholds measure local causal interoperability rather than grounding, phenomenal experience, or a complete theory of mathematical understanding. No model learned quantity through perception or action. A direct follow-up could train matched agents in a synthetic 2-D environment under three curricula: text-only descriptions, passive views of collections, and action-contingent tasks requiring agents to match, produce, and distribute n objects. Novel signs could be learned in disjoint contexts without direct pairing. Their later bidirectional reuse across new prompts and operations would test whether shared success conditions create the common numerical ground proposed here.

Acknowledgements

Generative AI tools assisted with literature search, coding, experiment orchestration, language editing of author-supplied material, and low-novelty drafting of factual Methods and Results descriptions from verified records. The substantive research questions, philosophical claims, and interpretations were supplied by the human author. Final verification, approval of every sentence, and responsibility for the submitted work remain with the human author. AI systems are not authors.

References

Lisa Feigenson, Stanislas Dehaene, and Elizabeth Spelke. 2004. Core systems of number. *Trends in Cognitive Sciences*, 8(7):307–314.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. 2025. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research*, 26(83):1–64.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1–3):335–346.

Hosein Hasani, Mohammadali Banayeeanzade, Ali Nafisi, Sadegh Mohammadian, Fatemeh Askari, Mobin Bagherian, Amirmohammad Izadi, and Mahdieh Soleymani Baghshah. 2026a. Mechanistic interpretability of large-scale counting in LLMs through a system-2 strategy. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 40885–40901. Association for Computational Linguistics.

Hosein Hasani, Amirmohammad Izadi, Fatemeh Askari, Mobin Bagherian, Sadegh Mohammadian, Mohammad Izadi, and Mahdieh Soleymani Baghshah. 2026b. Understanding counting mechanisms in large language and vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5125–5133.

George Lakoff and Rafael E. Núñez. 2000. *Where Mathematics Comes From: How the Embodied Mind Brings Mathematics into Being*. Basic Books, New York.

Tali Leibovich and Daniel Ansari. 2016. The symbol-grounding problem in numerical cognition: A review of theory, evidence, and outstanding questions. *Canadian Journal of Experimental Psychology*, 70(1):12–23.

Matthew Mandelkern and Tal Linzen. 2024. Do language models' words refer? *Computational Linguistics*, 50(3):1191–1200.

Andreas Nieder. 2021. Neuroethology of number sense across the animal kingdom. *Journal of Experimental Biology*, 224(6):jeb218289.

Michal Štefánik, Timothee Mickus, Marek Kadlčík, Bertram Højer, Michal Spiegel, Raúl Vázquez, Aman Sinha, Josef Kuchař, Philipp Mondorf, and Pontus

Stenetorp. 2026. Language models learn universal representations of numbers and here's why you should care. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30663–30681. Association for Computational Linguistics.

Bodo Winter and Jeff Yoshimi. 2020. Metaphor and the philosophical implications of embodied mathematics. *Frontiers in Psychology*, 11:569487.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Aojie Yuan and Zhiyuan Su. 2026. Beyond language: Format-agnostic reasoning subspaces in large language models. *Preprint*, arXiv:2605.09496. Preprint.

Fengting Yuchi, Li Du, and Jason Eisner. 2026. LLMs know more about numbers than they can say. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 659–673. Association for Computational Linguistics.

Definition Modeling for Mathematics: Understanding Audience Bias and its Problem Solving Relevance

Carolina Dias-Alexiou, Edison Marrese-Taylor, Alfredo Solano, Yutaka Matsuo
Graduate School of Engineering, The University of Tokyo
{carolina.dias,emarrese,asolano,matsuo}@weblab.t.u-tokyo.ac.jp

Abstract

Definition modeling is used as a probe of what a language model has internalized about a domain, and it has been studied on general lexicography and on jargon-heavy specialist domains. We extend this task to mathematics, where a term can be defined very differently for different communities. In this context we study the baseline capacity of models to retrieve generate definitions, and their audience-bias when explaining mathematical terms without specific guidance. We further asses to what extent can they follow a certain style when directed, and to which degree being exposed to definitions aid or hurt model performance when solving mathematical questions.

1 Introduction

A mathematician, a Wikipedia editor, and a general audience speaker, when asked to define a mathematical term like '*Klein bottle*' will likely give different definitions that can all be correct. Definitions are often not a single normalized strings per term, instead are shaped by readership and purpose, and we argue that is this of particular importance in mathematics. This makes such a domain a useful setting for separating whether a model "knows" a concept from whether it can explain it to a specified community.

Definition modeling was introduced as a probe of lexical representations (Noraset et al., 2017) and extended to contextual, multilingual, scientific, and specialist settings (Ishiwatari et al., 2019; August et al., 2022; Marrese-Taylor et al., 2025; Jhirad et al., 2023). We ask whether the same probe transfers to mathematics, where common concepts may be well represented in pretraining but their community-specific presentations differ sharply. The same definition bank also provides a direct test of definition-only retrieval: if the model was already exposed to a term, adding its definition may add distraction rather than information.

We align 2,963 headwords from the Concise Oxford Dictionary of Mathematics (Clapham and Nicholson, 2014) to entries on the English Wikipedia, the Simple English Wikipedia, nLab (a wiki for collaborative work on Mathematics, especially from the point of view of category theory), and the subreddit `r/explainlikeimfive`. We then two tasks designed to help shed light into the issue at hand: definition generation for both audience-agnostic and audience-tailored, and MMLU mathematics question answering with retrieved definitions. Our experiment utilize three different models: Qwen2.5-7B-Instruct, GPT-5-mini, and GPT-5.2.

2 Related Work

Definition modeling and controllable definitions. Definition modeling was introduced as a readout of lexical representations (Noraset et al., 2017) and surveyed by Gardner et al. (2022). Later work added contextual disambiguation (Gadetsky et al., 2018; Ishiwatari et al., 2019), generative dictionary systems (Periti et al., 2024; Tuan et al., 2024), scientific definitions with controllable complexity (August et al., 2022), and audience-controlled dictionary examples (He and Yiu, 2022). We extend this line to mathematical concepts and to five naturally occurring community registers rather than one complexity scale.

Knowledge and retrieval. Work on knowledge awareness asks whether a model can distinguish concepts it knows from concepts it does not (Ferrando et al., 2025); related work studies comprehension of newly emerging concepts under distribution shift (Mei et al., 2024). Retrieval-augmented generation (RAG) (Lewis et al., 2020) and selective variants (Asai et al., 2024; Jeong et al., 2024) likewise recognize that retrieved context is not always useful. We provide a controlled test in which retrieval is fixed and only definition text varies, con-

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 291–304
October 29, 2026 ©2026 Association for Computational Linguistics

Register	Readership	Terms	Med. len.
OXF	reference, terse	2,963	30
WIKI	general	2,229	160
SIMP	plain language	1,038	95
NLAB	research math	685	110
ELI5	lay, conversational	321	125

Table 1: The five-register corpus. Median length in words defines the audience-tailored length target.

necting term knowledge and MMLU mathematics performance.

3 Setup

3.1 Data

We begin with 2,963 headwords from the Concise Oxford Dictionary of Mathematics (OXF) (Clapham and Nicholson, 2014). For each headword, we collect independently authored text from four communities: the English Wikipedia article (general encyclopedic) denoted as WIKI, the Simple English Wikipedia (plain language) denoted as SIMP, nLab (research mathematics with a category-theoretic perspective) referred to as NLAB, and an the content of the subreddit `r/explainlikeimfive` answers (lay, conversational), denoted ELI5.

Starting from OXF as a fixed vocabulary of terms, we built parallel term-aligned corpora against the other sources via a candidate-generation and LLM-adjudication pipeline. For each source, candidates were first retrieved with a lightweight, source-appropriate method. An LLM agent then judged, per term, whether any candidate named the same mathematical concept, applying source-specific disambiguation rule.

For WIKI, SIMP and NLAB, this judgment was followed by a second verification stage in which a another agent re-adjudicated each accepted pair, now grounded in the candidate's actual lead text. Finally, because a Reddit thread's title is uninformative about its content, its single judgment stage instead conditioned directly on the thread's answer text and emitted a high/medium/low confidence label rather than going through a separate verification pass. Only confirmed/related pairs from each source were retained and joined back onto the Oxford vocabulary by term to produce the final parallel corpus. Please see §F for details.

Table 1 summarizes the details of the data we collected and utilized. Length values reported in the table for WIKI and SIMP are only for the first paragraph of the article, which is what we employ for the main experiments in this paper, unless otherwise specified. Even under this scenario, we note that length and technicality/complexity are not strictly correlated. We see that both the nLab and the subreddit entries have a very similar length (110 and 125 words respectively) albeit having almost opposite audience as targets. We therefore avoid interpreting the registers as a single complexity scale.

Because term coverage depends on the source, our common-term analyses use explicit intersections. The *three-way* core contains 957 terms with Oxford, Wikipedia, and Simple Wikipedia text, and the *five-way* core contains 97 terms with text in all five registers. Please see §3.2 for details.

3.2 Tasks

Definition modeling. We begin by testing a panel of models on the task of definition modeling in this domain via our collected data. We evaluate *Qwen2.5-7B-Instruct*, *GPT-5-mini*, *and GPT-5.2*. A bare-term prompt asks for a two-to-three-sentence audience-agnostic definition. In a second, tailored condition, the prompt names the target readership and gives its median-derived word-count target. All generations are scored against each gold register using BLEU, METEOR, COMET, and Baseline-rescaled BERTScore, plus an LLM-as-a-Judge approach based on GPT-5.2 and Gemini-3.5-Flash for correctness and register-fit on a 0–3 scale.

MMLU mathematics performance. We use 1,189 test questions from six MMLU mathematics subjects (Hendrycks et al., 2020) to assess whether having access to gold-standard definitions helps models solve the questions correctly. Since there is no gold-standard dataset of question-term relations for MMLU, we fuse an TF-IDF and dense retrieval approach with a cross-encoder re-ranker to produce ten candidate terms related to each question. The term-question mapping was derived independently by two LLM judges over all 1,189 questions, each verifying every selected term against the same three-criterion rubric (definitional support, necessity, semantic role) and flagging required terms missed among the rejected candidates. For this we utilized GPT-5.2 (orig.), which we also compare against Gemini-3.5-Flash (Judge A) and GPT-5-mini (Judge B). Table 2 shows the details.

As shown in the table, both additional judges

Comparison	EM	Jac.	κ	P	R
orig. vs. judge A	70.9	0.772	0.840	0.968	0.773
orig. vs. judge B	51.1	0.599	0.629	0.911	0.525
judge A vs. judge B	61.7	0.702	0.680	—	—

Table 2: Validation of question-to-term mapping, where EM stands for Exact Match, Jac. is short for Jaccard Index, κ indicates Cohen's Kappa, and P and R are short for Precision and Recall, computed assuming GPT-5.2 as gold standard.

reach high agreement with the original annotation, largely by confirming its selections (precision $\approx$ 0.9) while trimming about a quarter of them. Judge B applies a stricter necessity criterion, removing roughly half of the original's picks. While GPT-5.2 and Gemini-3.5-Flash have both high Jaccard Index ($0.736 \sim 0.834$) and high Exact Match ($69.1\% \sim 72.9\%$) for all subjects, GPT-5-mini has a highest disagreement with GPT-5.2. Its per-subject Jaccard is highest on `high_school_statistics` (0.666) and `elementary_mathematics` (0.648) and lowest on `abstract_algebra` (0.540) and `college_mathematics` (0.485), but more striking is the difference in the Exact match which drops to 35.7% and 35.0% for `formal_logic` and `college_mathematics` respectively. The mapping final averages 1.37 terms per question, with 230 no-term questions. Although we did not test this empirically, we believe this suggest results would not change substantially when using a similarly-capable model for this kind of annotation.

For the RAG stage, retrieval is run once over the OXF text and the same top-three term identifiers are reused in every condition. Only the injected text changes: none, Oxford, Wikipedia lead, or Simple Wikipedia. This isolates definition register from retrieval quality. As mentioned earlier, the full Wikipedia article was tested separately, please see below.

4 Results

4.1 Definition Modeling

Table 3 summarizes our results on the definition modeling experiments. Although all entries have relatively low values (non greater than 5), WIKI shows the highest BLEU scores, especially for Qwen2.5-7B and GPT-5-mini. This is true for both agnostic and tailored answers. For the latter, even GPT-5.2 shows a relatively high value of BLEU. This could indicate a bias in terms of model exposure during training. We also noticed that Qwen2.5-7B retains the strongest BERTScore in every register.

Comparing between agnostic and tailored scenarios, Qwen2.5-7B on NLAB correctness falls to $0.99/3$ over 684 tailored terms, well below its own general NLAB score, while retaining near-ceiling register-fit on ELI5 (2.98) despite the panel's lowest correctness there. We think this is not a knowledge gap since, when scored under the plain, audience-agnostic prompt on the full 684-term NLAB set, Qwen2.5-7B scores $2.32/3$. The failure appears only once the prompt asks it to write in the NLAB register.

Since the correctness judge is GPT-5.2, one of the three models it scores, we independently re-judged all (model, register, term) triples with Gemini-3.5-Flash. The two judges exactly agree on 79.6% of items, with Cohen's $\kappa = 0.313$ (unordered categories) and Kendall's $\tau = 0.487$. Gemini is uniformly more lenient (mean score 2.88 vs. 2.66), concentrated in upgrading a 2 to a 3 rather than reversing outright failures, so the two judges rarely disagree about which generations are broken, only about how generously to score borderline ones. Please see detailed results in §A.

Table 4 shows the comparison of each tailored generation against all five golds on the terms for which we have definitions at all levels (the *five-way* core) using the correctness measure. Rows are the register the generation was written for; columns are the gold it is scored against.

The most immediate observation is that there is little variation between the values, although with a few exceptions. This shows a potential limitation our LLM-as-a-Judge evaluation, since one would except values to be more concentrated/higher for a single audience. We think this could be an indication that our tested models are not sufficiently good at making this kind of judgments, but we reckon that it can also be due our limited budget in terms of prompt design and optimization.

Overall, we see tailoring to a named audience is close to free on OXFORD and WIKIPEDIA ($|\Delta\text{Corr}| \leq 0.16$ and ≤ 0.06, respectively) and increasingly costly moving to SIMPLE (-0.19 to -0.54), ELI5 (-0.45 to -0.83), and NLAB (-0.27 to -1.33). The losses are generally smallest for the two API models and largest for Qwen2.5-7B.

When limited to cross-register scoring on the *five-way* core, we can see more clearly that the

		Agnostic			Tailored			
Reg.	**Model**	**BL**	**BS**	**Corr.**	**BL**	**BS**	**Corr.**	**Reg**
OXF	Qwen2.5-7B	1.78	5.59	2.44	1.24	6.97	2.34	2.26
	GPT-5-mini	1.36	0.91	2.64	0.50	-3.96	2.49	2.71
	GPT-5.2	1.00	1.34	2.63	0.30	-4.33	2.55	2.80
WIKI	Qwen2.5-7B	3.43	4.08	2.59	4.55	4.42	2.50	2.70
	GPT-5-mini	3.40	-1.70	2.82	4.63	2.88	2.78	2.92
	GPT-5.2	1.64	-6.23	2.80	3.09	0.91	2.78	2.99
SIMP	Qwen2.5-7B	3.22	5.17	2.67	1.96	0.27	2.13	2.90
	GPT-5-mini	2.56	-2.66	2.81	2.59	2.12	2.63	2.78
	GPT-5.2	1.31	-5.76	2.81	1.68	1.08	2.53	2.93
NLAB	Qwen2.5-7B	1.37	0.54	2.32	0.81	-1.90	0.99	1.90
	GPT-5-mini	1.62	-1.60	2.65	1.09	-10.53	2.10	2.88
	GPT-5.2	0.61	0.57	2.59	0.53	-7.14	2.31	2.94
ELI5	Qwen2.5-7B	1.42	2.43	2.82	0.77	0.27	1.99	2.98
	GPT-5-mini	1.29	-3.19	2.95	0.70	-1.89	2.50	3.00
	GPT-5.2	0.55	-7.06	2.94	0.60	-2.78	2.48	3.00

Table 3: BLEU (BL), Baseline-rescaled BERTScore (BS, all $\times 100$), and Correctness and tailored Register-fit (Reg.) via GPT-5.2 for audience-agnostic and audience-tailored generation; judge scores are $0 \sim 3$. Scores are computed over each register's native available term set. Baseline-rescaled BERTScore may be negative.

	Model	OXF	WIKI	SIMP	NLAB	ELI5
OXFORD	Qwen2.5-7B	**2.71**	2.67	2.77	2.55	2.81
	GPT-5-mini	**2.81**	2.86	2.84	2.72	2.87
	GPT-5.2	**2.86**	2.88	2.90	2.70	2.90
WIKIPEDIA	Qwen2.5-7B	2.43	**2.61**	2.66	2.30	2.70
	GPT-5-mini	2.75	**2.81**	2.88	2.56	2.89
	GPT-5.2	2.64	**2.76**	2.79	2.45	2.78
SIMPLE	Qwen2.5-7B	1.98	2.07	**2.25**	1.93	2.20
	GPT-5-mini	2.44	2.60	**2.70**	2.26	2.74
	GPT-5.2	2.42	2.54	**2.68**	2.25	2.68
NLAB	Qwen2.5-7B	1.08	1.18	1.19	**1.21**	1.15
	GPT-5-mini	2.14	2.21	2.16	**2.24**	2.25
	GPT-5.2	2.20	2.36	2.31	**2.36**	2.35
ELI5	Qwen2.5-7B	1.65	1.65	1.86	1.55	**1.76**
	GPT-5-mini	1.98	2.09	2.18	1.80	**2.15**
	GPT-5.2	1.93	1.98	2.07	1.76	**2.15**

Table 4: Correctness judge ($0 \sim 3$) for every (written-for, scored-against) register pair on the *five-core* (n=97). Bold marks the Correctness judgment for the matching target and underline marks the uniformly worst row.

Model	**Cohort**	**Base**	OXF	WIKI	SIMP
Qwen2.5-7B	all	53.32	−0.17	−1.09	−2.27
	has	49.73	+0.91	+2.37	−1.28
	no	56.41	−1.09	−4.06*	−3.12
GPT-5-mini	all	65.01	−1.09	−0.76	−1.35
	has	64.30	−1.64	−1.82	−3.64*
	no	65.62	−0.62	+0.16	+0.62
GPT-5.2	all	74.26	−0.42	−0.08	−0.17
	has	72.86	−1.09	+0.73	−0.73
	no	75.47	+0.16	−0.78	+0.31

Table 5: MMLU-math accuracy change (percentage points) from injecting a retrieved definition. Retrieval is fixed. Cohorts are all ($n = 1{,}189$), has-term ($n = 549$), and no-term ($n = 640$). * marks a paired-bootstrap 95% CI excluding zero; all such effects are losses.

NLAB-tailored generation itself seems to be the reason of the failure: Qwen2.5-7B's NLAB-tailored row is the worst row of the matrix against every gold, not only against NLAB's own.

We inspected the low-scoring outputs to better understand why this may happen. We found that for every one of the 547 terms whose gold definition uses no categorical-theoretic vocabulary, Qwen2.5-7B's nLab-tailored generation nonetheless forces one, e.g. defining *acceleration* as a "2-morphism between functors" rather than the "second derivative of a curve". The model over-applies the requested register rather than omitting content it lacks. An explicit guardrail against inventing structure recovered over half of the gap (correctness $0.99 \rightarrow 1.57$) without a compensating drop in register-fit, and a held-out real exemplar gives a smaller, Qwen-specific gain ($0.99 \rightarrow 1.18$); neither fully restores plain-prompt performance (please see §B for details).

4.2 MMLU Mathematics Performance

For our RAG experiments, against the three-way corpus (n=1,189), we split questions into two cohorts: *has-term* (n=549) questions, whose required term falls inside that corpus, so the definition retrieved and injected is genuinely the one the question needs; and *no-term* (n=640) questions, for which any injected definition is necessarily off-target. This split lets us test whether RAG helps specifically where its content is relevant and only distracts where it is not.

As shown in Table 5, no model gains significantly from any short definition register. Qwen2.5-7B has the arguably expected split on relevant versus irrelevant questions, but only the irrelevant WIKI condition survives the bootstrap test (-4.06 points, 95% CI $[-7.66, -0.62]$). GPT-5-mini is significantly harmed on has-term questions by SIMPLE; GPT-5.2 is statistically flat. Thus, relevance alone does not guarantee benefit. In this benchmark, correct definitions are usually redundant and can become distraction.

We also observed that the effect is subject-dependent. Qwen2.5-7B benefits most in concept-dense subjects such as abstract algebra and for-

mal logic, while all three short-definition registers hurt elementary mathematics. Manual inspection suggests that the latter questions often mention a related operation without requiring its definition, exposing a limitation of the LLM term-necessity annotation (full breakdown in § D).

Finally, more context is substantially worse for the smaller model. Replacing the roughly 1,200-character WIKIPEDIA lead with the entire article changes Qwen's accuracy by -9.34 points overall, while GPT-5-mini loses 1.93 points. Raw-output tracing shows genuine incoherent multilingual generation under the extended context, not a scoring artifact. For Qwen, this significantly negative effect may be due to enabling extended context via YaRN scaling (Peng et al., 2024).

5 Conclusion

We extended definition modeling to mathematics with five community registers and paired it with definition-only retrieval for MMLU mathematics. Surface metrics and correctness judges disagree systematically; Qwen's nLab failure is a register-conditioning instruction-following problem rather than a knowledge gap, and is largely, though not fully, corrected by explicit prompt guardrails; and short retrieved definitions produce no reliable gains, while excessive context causes substantial degradation. These results argue for multi-register evaluation and targeted, compact context rather than treating one reference definition or more retrieved text as universally helpful.

Limitations

Self-judging. We GPT-5.2 as a judge for both experiments, but this is also one of the three models being evaluated, so a third of every judged comparison is a self-assessment. The judge-score band across models is narrow (2.63–2.94 audience-agnostic), and we cannot rule out that leniency toward its own and its sibling's outputs contributes to it.

The reference-based metrics are judge-independent and show the same model ordering, and Qwen2.5-7B's NLAB failure. These results show that GPT-5.2 has no incentive to favor is additionally corroborated by a directly inspectable mechanism. A replication with a judge outside the panel would nonetheless be worth running.

In our experiments, each audience instruction is paired with a word-count target derived from that register's own gold median, so the tailored generations differ from the audience-agnostic ones in both register and length. We therefore cannot separate the instruction effect from the length constraint.

Only 23% of headwords have an NLAB page and 11% an ELI5 thread, and neither subset is random: NLAB skews toward algebraic and structural mathematics, ELI5 toward terms popular enough to be asked about on a general forum. Comparisons across registers are therefore comparisons across different, non-exchangeable term sets, except on the 97-term five-way core.

Three models is a thin panel, with a single open model at 7B and no mid-scale open models between it and the two API models, so we cannot tell whether Qwen2.5-7B's NLAB failure is a scale effect, a Qwen-family effect, or specific to this checkpoint.. Furthermore, the retrieval deltas in §4.2 are small relative to sampling error at n=549–1,189: we report which ones survive a paired bootstrap and confine our claims to those, but the study is not powered to detect gains below roughly 3pp at the cohort level.

Finally, the term-question mapping is produced by an LLM annotator and validated only by two further LLM judges (Table 2); no human review of the 1,189-question mapping has yet been completed, and the `elementary_mathematics` evidence in §4.2 suggests the necessity criterion is looser than intended in the procedural regime.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Tal August, Katharina Reinecke, and Noah A. Smith. 2022. Generating scientific definitions with controllable complexity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8298–8317.

Christopher Clapham and James Nicholson. 2014. *The Concise Oxford Dictionary of Mathematics*, 5 edition. Oxford University Press.

Javier Ferrando, Oscar Balcells Obeso, Senthooran Rajamanoharan, and Neel Nanda. 2025. Do i know this entity? knowledge awareness and hallucinations in language models. In *The Thirteenth International Conference on Learning Representations*.

Artyom Gadetsky, Ilya Yakubovskiy, and Dmitry Vetrov. 2018. Conditional generators of words definitions.

In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 266–271.

Noah Gardner, Hafiz Khan, and Chih-Cheng Hung. 2022. Definition modeling: literature review and dataset analysis. *Applied Computing and Intelligence*, 2(1):83–98.

Xingwei He and Siu Ming Yiu. 2022. Controllable dictionary example generation: Generating example sentences for specific targeted audiences. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 610–627.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Shonosuke Ishiwatari, Hiroaki Hayashi, Naoki Yoshinaga, Graham Neubig, Shoetsu Sato, Masashi Toyoda, and Masaru Kitsuregawa. 2019. Learning to describe unknown phrases with local and global contexts. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3467–3476.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 7036–7050.

James Jhirad, Edison Marrese-Taylor, and Yutaka Matsuo. 2023. Evaluating large language models' understanding of financial terminology via definition modeling. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 93–100.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Edison Marrese-Taylor, Erica K. Shimomoto, Alfredo Solano, and Enrique Reid. 2025. Multilingual definition modeling. In *Findings of the Association for Computational Linguistics: ACL 2025*.

Lingrui Mei, Shenghua Liu, Yiwei Wang, Baolong Bi, and Xueqi Cheng. 2024. SLANG: New concept comprehension of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12558–12575.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2017. Definition modeling: Learning to define word embeddings in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*.

Francesco Periti, David Alfter, and Nina Tahmasebi. 2024. Automatically generated definitions and their utility for modeling word meaning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Kai-Wen Tuan, Hai-Lun Tu, and Jason S. Chang. 2024. Generative dictionary: Improving language learner understanding with contextual definitions. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 390–396.

A Full Definition-Modeling Results

Table 6 reports all reference metrics (BLEU, METEOR, COMET, BERTScore) alongside both correctness judges (GPT-5.2 and `gemini-3.5-flash`, Corr_G) for the audience-agnostic condition, and all reference metrics alongside correctness and register-fit for the audience-tailored condition, expanding the summary in Table 3.

B Audience Instructions

Each register's tailored prompt appends one instruction and one length target (Table 1) to the bare-term prompt. Abridged: OXFORD — "precise, formal, dictionary-style, for a reader with mathematical training; be terse and exact; do not simplify, add examples, or add motivation". WIKIPEDIA — "standard encyclopedia-style explanation for an educated general reader with no specialized mathematics background, like a Wikipedia opening paragraph: define the term and give brief context". SIMPLE — "simple explanation for a non-specialist or early learner; plain vocabulary and short sentences; avoid jargon". NLAB — "rigorous, structural explanation for a research mathematician familiar with category theory; use categorical or structural language and, where natural, situate the notion in a more general setting". ELI5 — "conversational, intuitive explanation for a curious adult with no mathematics background, in the style of an `r/explainlikeimfive` answer; use everyday language and an analogy if it helps; avoid equations and jargon".

The one-shot exemplar for each register is the gold entry whose length is closest to that register's median, excluding the five-way core; it is held out from that register's evaluation set.

C Retrieval Confidence Intervals

Table 9 gives the paired-bootstrap 95% CIs (2,000 resamples over per-question correctness, resampling questions so that the baseline and RAG runs stay paired) behind Table 5.

D Per-Subject Breakdown

Table 10 breaks Qwen2.5-7B's has-term cohort down by MMLU subject. Cell sizes are 63–120 questions, so individual deltas are exploratory; the pattern of interest is that `elementary_mathematics` is negative at all three registers while the concept-dense, low-baseline subjects are mostly positive.

E Infrastructure Notes

Reasoning-effort truncation. GPT-5-mini defaults its hidden reasoning effort to a level that, on short-answer prompts with a modest output budget, consumes the entire budget before emitting any visible text; the response returns incomplete and empty. Left unfixed this silently collapsed every prediction to the first option and produced a spurious 21% baseline. We pin the lowest effort level each variant supports and raise the output-token floor for all API calls in both studies.

Long-context serving. The full-article condition requires more than Qwen2.5-7B's native 32K context; we serve it with static YaRN scaling (Peng et al., 2024) at 131K. No article needed truncation at that budget (longest ≈351K characters). vLLM's echo-plus-logprob scoring path fails server-side at these lengths with a non-finite value error in both fp16 and bf16, so all full-article runs score multiple-choice questions by generation and letter extraction instead of per-option logprobs.

F Term Matching

F.1 English Wikipedia

Candidates were generated per letter by fuzzy-matching each Oxford term (via difflib.SequenceMatcher) against Wikipedia's mathematics-article title list, keeping the top 5. An LLM judge then picked the single best title-only match using explicit disambiguation rules (rejecting narrower/broader pages and cross-subfield homonyms), after which a second agent re-verified each accepted pair against the article's real fetched lead paragraph, producing a confirmed/related/wrong verdict. Any pair left unconfirmed went through a repair stage in which a tool-using agent actively searched Wikipedia for a better target article before giving up.

For the case of the Simple English Wikipedia, this structurally identical to the above. We use same fuzzy-matching candidate generation and the exact same judge and verify prompts — with only the target host swapped (simple.wikipedia.org). However, no repair/repoint stage was run for this source.

F.2 nLab

Since the nLab exposes no search API, candidates were generated by matching each Oxford term locally against the wiki's complete page-name index,

		Agnostic						Tailored					
Register	**Model**	**BL**	**MET**	**COM**	**BS**	**Corr**	**Corr$_G$**	**BL**	**MET**	**COM**	**BS**	**Corr**	**Reg**
OXF	Qwen2.5-7B	1.78	18.11	56.96	5.59	2.44	2.72	1.24	13.60	54.15	6.97	2.34	2.26
	GPT-5-mini	1.36	17.89	55.64	0.91	2.64	2.85	0.50	12.02	51.24	-3.96	2.49	2.71
	GPT-5.2	1.00	15.90	56.67	1.34	2.63	2.86	0.30	9.70	51.29	-4.33	2.55	2.80
WIKI	Qwen2.5-7B	3.43	16.66	62.04	4.08	2.59	2.85	4.55	22.72	68.90	4.42	2.50	2.70
	GPT-5-mini	3.40	17.55	60.32	-1.70	2.82	2.95	4.63	24.68	69.62	2.88	2.78	2.92
	GPT-5.2	1.64	13.56	58.06	-6.23	2.80	2.96	3.09	23.24	69.72	0.91	2.78	2.99
SIMP	Qwen2.5-7B	3.22	18.19	62.90	5.17	2.67	2.92	1.96	16.57	58.90	0.27	2.13	2.90
	GPT-5-mini	2.56	18.29	59.55	-2.66	2.81	2.96	2.59	20.63	63.40	2.12	2.63	2.78
	GPT-5.2	1.31	14.56	58.38	-5.76	2.81	2.97	1.68	18.66	62.38	1.08	2.53	2.93
NLAB	Qwen2.5-7B	1.37	14.74	57.11	0.54	2.32	2.75	0.81	16.27	59.66	-1.90	0.99	1.90
	GPT-5-mini	1.62	15.97	57.02	-1.60	2.65	2.94	1.09	17.18	54.91	-10.53	2.10	2.88
	GPT-5.2	0.61	12.90	59.04	0.57	2.59	2.93	0.53	14.74	56.24	-7.14	2.31	2.94
ELI5	Qwen2.5-7B	1.42	12.81	58.22	2.43	2.82	2.98	0.77	16.69	58.18	0.27	1.99	2.98
	GPT-5-mini	1.29	13.31	57.02	-3.19	2.95	2.98	0.70	18.03	58.71	-1.89	2.50	3.00
	GPT-5.2	0.55	10.80	55.35	-7.06	2.94	2.98	0.60	16.61	59.52	-2.78	2.48	3.00

Table 6: BLEU (BL), METEOR (MET), COMET (COM), BERTScore (BS, all $\times 100$), correctness (Corr), and tailored register-fit (Reg) for audience-agnostic and audience-tailored generation, judged at a 2,000-character cap; judge scores are 0–3. Corr is judged by GPT-5.2; Corr$_G$ is the same agnostic generations judged by `gemini-3.5-flash` (§4.1). Scores use native available term sets; nLab and ELI5 tailored runs contain 684 and 320 scored terms because one term per register is held out for the one-shot ablation. Negative BERTScore values arise from baseline rescaling.

Model / tier	**Plain**	**Guarded**	**Scaffold**
Qwen2.5-7B / NLAB	0.99 / 1.90	**1.57 / 1.85**	1.52 / 1.84
GPT-5-mini / NLAB	2.07 / 2.89	**2.49 / 2.54**	2.29 / 2.94
GPT-5.2 / NLAB	2.30 / 2.92	**2.67 / 2.49**	2.46 / 2.97
Qwen2.5-7B / ELI5	1.99 / 2.98	2.17 / **2.96**	**2.26 / 2.94**
GPT-5-mini / ELI5	2.52 / 3.00	2.70 / 2.92	2.63 / 2.98
GPT-5.2 / ELI5	2.54 / 3.00	**2.73 / 2.96**	2.58 / 3.00

Table 7: Correctness / register-fit (0–3) for the plain audience-tailored prompt versus two guardrail variants, at the 2,000-character cap. Guarded forbids inventing structure the term does not need; scaffold requires one plain content sentence first. Neither fully restores plain-prompt (audience-agnostic) correctness, but guarded closes most of Qwen2.5-7B's NLAB gap without a register-fit cost.

Model / NLAB	**Corr. 0→1-shot**	**Reg. 0→1-shot**
Qwen2.5-7B	0.99 → **1.18 (+0.19)**	1.90 → 1.90 (flat)
GPT-5-mini	2.10 → 2.13 (+0.03)	2.88 → 2.87 (flat)
GPT-5.2	2.31 → 2.25 (−0.06)	2.94 → 2.91 (flat)

Table 8: One-shot exemplar ablation on NLAB (n=684). The exemplar helps only Qwen2.5-7B, and only content, not register-fit; a first release of this ablation used the corrupted exemplar at the 500-character cap and reported an inflated +0.33 for Qwen2.5-7B, which this table supersedes.

then probing terms with no match via HEAD requests to recover synonym redirects the index does not list. An LLM judge picked the best candidate from page names alone, using rules tuned to the nLab's category-theoretic style (a more abstract page can still be the right match) and its main failure mode (a same-cluster redirect, e.g. "ceiling", "floor", is not a match). A second agent then re-verified each match against the page's real fetched prose, with added handling for biography pages and content-free "stub" pages.

F.3 Redddit ELI5

Candidates were found by searching Reddit per term and keeping only threads where the term appears as a whole phrase in the question title, restricted by default to Mathematics-flaired threads for precision. Because a thread's title is a question rather than a description, the judge conditioned directly on the thread's fetched best-answer text rather than the title, and — since that already grounds the judgment in real content — this source has no separate verification stage; the judge instead emits a high/medium/low confidence label in place of a confirmed/related/wrong verdict.

Model	Injected	all (n=1,189)	has (n=549)	no (n=640)
Qwen2.5-7B	OXFORD	-0.17 [-2.27, $+1.93$]	$+0.91$ [-1.82, $+3.83$]	-1.09 [-4.06, $+2.03$]
	WIKIPEDIA	-1.09 [-3.62, $+1.43$]	$+2.37$ [-1.09, $+5.83$]	-4.06^{*} [-7.66, -0.62]
	SIMPLE	-2.27 [-4.88, $+0.17$]	-1.28 [-4.74, $+2.19$]	-3.12 [-6.72, $+0.16$]
	WIKIPEDIA $_{F}$	-9.34^{*} [-12.70, -5.80]	-4.37 [-9.47, $+1.09$]	-13.59^{*} [-18.13, -8.75]
GPT-5-mini	OXFORD	-1.09 [-3.20, $+0.93$]	-1.64 [-4.92, $+1.46$]	-0.62 [-3.28, $+2.19$]
	WIKIPEDIA	-0.76 [-2.94, $+1.35$]	-1.82 [-4.74, $+1.28$]	$+0.16$ [-2.97, $+2.97$]
	SIMPLE	-1.35 [-3.36, $+0.59$]	-3.64^{*} [-6.56, -0.72]	$+0.62$ [-2.19, $+3.28$]
	WIKIPEDIA $_{F}$	-1.93 [-3.95, $+0.17$]	-3.64^{*} [-6.74, -0.73]	-0.47 [-3.12, $+2.19$]
GPT-5.2	OXFORD	-0.42 [-2.35, $+1.51$]	-1.09 [-4.01, $+1.82$]	$+0.16$ [-2.34, $+2.50$]
	WIKIPEDIA	-0.08 [-2.02, $+1.85$]	$+0.73$ [-2.19, $+3.83$]	-0.78 [-3.44, $+1.88$]
	SIMPLE	-0.17 [-2.02, $+1.85$]	-0.73 [-3.64, $+2.37$]	$+0.31$ [-2.34, $+2.97$]

Table 9: Accuracy change (pp) with paired-bootstrap 95% CIs. * marks CIs excluding zero. No positive delta in the table is distinguishable from zero.

Subject	n	Base	OXF	WIKI	SIMP
abstract algebra	66	37.9	+9.09	+9.09	−7.58
college maths	63	38.1	−3.17	+3.17	−4.76
formal logic	84	41.7	+1.19	+5.95	+4.76
HS maths	102	51.0	+0.98	+3.92	−1.96
HS statistics	120	52.5	+1.67	+2.50	+5.00
elementary maths	114	64.9	−2.63	−6.14	−6.14

Table 10: Qwen2.5-7B, has-term cohort only, accuracy change (pp) by subject, ordered by baseline accuracy.

Subject	n	Jaccard	Exact match
formal_logic	126	0.834	69.1%
college_mathematics	100	0.812	71.0%
high_school_statistics	216	0.795	71.3%
abstract_algebra	100	0.783	70.0%
elementary_mathematics	377	0.751	72.9%
high_school_mathematics	270	0.736	68.9%

Table 11: GPT-5.2 (original) vs. Gemini judge — overall $\kappa = 0.840$, Jaccard $= 0.772$, exact match $= 70.9\%$

Subject	n	Jaccard	Exact match
elementary_mathematics	377	0.648	62.9%
high_school_statistics	216	0.666	53.7%
formal_logic	126	0.584	35.7%
high_school_mathematics	270	0.548	49.6%
abstract_algebra	100	0.540	40.0%
college_mathematics	100	0.485	35.0%

Table 12: GPT-5.2 (original) vs. gpt-5-mini judge — overall $\kappa = 0.629$, Jaccard $= 0.599$, exact match $= 51.1\%$

Subject	n	Jaccard	Exact match
elementary_mathematics	377	0.819	79.8%
high_school_mathematics	270	0.701	65.2%
high_school_statistics	216	0.683	56.0%
formal_logic	126	0.606	40.5%
abstract_algebra	100	0.597	45.0%
college_mathematics	100	0.529	40.0%

Table 13: Gemini vs. gpt-5-mini judge — overall $\kappa = 0.680$, Jaccard $= 0.702$, exact match $= 61.7\%$

Wikipedia Judge Prompt (Stage 1)

```
You are matching dictionary entries from the Oxford Dictionary of Mathematics to Wikipedia
↪ article titles. Decide, for each dictionary entry, whether any candidate Wikipedia title
↪ refers to the SAME mathematical topic/concept.
Read this JSON file: ${infile}
Each element has "term", "definition" (may be truncated with ...), and "candidates" (up to 5
↪ Wikipedia titles with fuzzy scores you may ignore).
For each entry pick the ONE candidate that refers to the same topic as term+definition, or null
↪ if none does.
Rules for "same topic":
- Singular/plural, word order, and disambiguation suffixes like "(mathematics)" DO count as the
↪ same topic (e.g. "antipodal points" == "Antipodal point").
- A page whose title CONTAINS the term but is about a narrower/different concept is NOT a match (
↪ e.g. "angle" vs "Angle trisection"; "algebra" vs "Azumaya algebra"; "abacus" vs "Abacus school
↪ ").
- A broader page counts ONLY if it is genuinely the primary article for the term (e.g. "analysis"
↪  == "Mathematical analysis"); "affine geometry" vs "Geometry" is too broad, NOT a match.
- Biographies of mathematicians match their person page, but not an unrelated person or a journal/
↪ organization.
- Beware false friends: the same word can name different concepts in different subfields -- use
↪ the definition to disambiguate.
- When in doubt, prefer null over a wrong match.
Write a JSON array to EXACTLY this path: ${outfile}
Each element: {"term": <term>, "match": <chosen Wikipedia title string or null>, "confidence": "
↪ high"|"medium"|"low", "reason": <short phrase>}
Keep every input term, in order. After writing the file, reply with just the count of non-null
↪ matches.
```

Figure 1: Stage-1 judge prompt for the English Wikipedia source. The judge sees only candidate article *titles* (no article text) and picks the single best title-only match, or `null`. Reused verbatim, with host swapped, for Simple English Wikipedia.

Wikipedia Verification Prompt (Stage 2)

```
You are verifying candidate matches between Oxford Dictionary of Mathematics entries and
↪ Wikipedia articles, using the ACTUAL Wikipedia article intro text.
Read this JSON file: ${infile}
Each element has "term", "definition" (full Oxford definition), "wikipedia_title", "wiki_intro" (
↪ actual lead paragraph(s)), "prior_confidence" (ignore).
For each element decide whether the Wikipedia article (per its intro) describes the SAME
↪ mathematical concept as the dictionary definition.
Verdicts:
- "confirmed" - clearly the same concept.
- "related" - overlapping/adjacent but NOT the same concept (a sub-case, a broader field, a
↪ different-but-related notion, or a disambiguation page).
- "wrong" - different topic / wrong sense / homonym collision.
Judge on MEANING using both the definition and the intro, not on title spelling. Watch for false
↪ friends (same word, different concept in a different subfield). Singular/plural and
↪ disambiguation suffixes are still the same concept.
Write a JSON array to EXACTLY this path: ${outfile} Each element: {"term": <term>, "
↪ wikipedia_title": <title>, "verdict": "confirmed"|"related"|"wrong", "note": <short reason
↪ grounded in the intro>}
Keep all input elements, in order. After writing, reply with just a tally "confirmed=X related=Y
↪ wrong=Z".
```

Figure 2: Stage-2 verification prompt for the English Wikipedia source. A fresh agent re-adjudicates each Stage-1 match, now grounded in the article's real fetched lead text. Also reused verbatim for Simple English Wikipedia.

Wikipedia Repoint Prompt (Stage 3)

```
You are fixing one dictionary->Wikipedia match that was judged NOT to be the same topic. Find a
↪ better Wikipedia article for it, if one exists. Target wiki host: ${host}.
Step 1. Read item index ${i} (0-indexed) of this JSON array file: ${pipe}/repoint_todo.json
It has: "term", "definition" (full Oxford definition), "current_title" (the REJECTED article), "
↪ verdict", "note".
Step 2. Search Wikipedia for the article that describes the SAME concept as term+definition. Run
↪ this helper (it prints JSON [{title, resolved, intro, missing}]):
    python3 ${oxdir}/wiki_search.py --host ${host} "your search query"
Try 1-3 queries using the term and key words from the definition. You may also check specific
↪ titles:
    python3 ${oxdir}/wiki_search.py --host ${host} --titles "Some Title" "Another Title"
Step 3. Decide, grounded in the fetched intros:
- If some article clearly describes the SAME concept (exact concept, or the primary article for
↪ it), choose it. Disambiguation suffixes and singular/plural are fine.
- If the best you can find is still only broader/narrower/related, or the concept has no
↪ standalone article (e.g. it only redirects to a disambiguation page or is a mere section),
↪ choose NONE.
Step 4. Write a JSON object to EXACTLY this path: ${pipe}/repoint_out/${i}.json
Shape: {"term": <term>, "current_title": <the rejected title>, "new_title": <the better article
↪ title, or null if none exists>, "verdict": "confirmed" if you found a same-concept article
↪ else "no_better_target", "note": <short reason grounded in the intro you read>}
After writing, reply with one line: term -> new_title (or "no better target").
```

Figure 3: Repair-loop prompt run only over English Wikipedia pairs left unconfirmed after Stage 2. Unlike the judge/verify prompts, this agent is tool-using: it actively issues search queries against `wiki_search.py` before deciding on a replacement target. No analogous stage exists for the other three sources.

nLab Judge Prompt (Stage 1)

```
You are matching entries from the Oxford Dictionary of Mathematics to pages on the nLab (https://
↪ ncatlab.org/nlab/), a category-theory/higher-mathematics wiki.
Read this JSON file: {infile}
Each element has "term", "definition" (the Oxford definition, may be truncated with ..., and "
↪ candidates": nLab page names. Each candidate has a "source" field ("name" = matched by page
↪ name, "redirect" = the nLab redirects this term to that page) which you may use as weak
↪ context but should not treat as evidence of correctness -- redirect candidates in particular
↪ are often a related-but-different concept. Some entries have only one candidate.
For each entry pick the ONE candidate naming the same mathematical concept as term+definition, or
↪  null if none does. Judge on MEANING using the definition, not on title spelling. Rules for "
↪ same topic":
- Singular/plural, word order, hyphenation, British/American spelling, and disambiguation
↪ suffixes DO count as the same topic ("zero divisors" == "zero-divisor", "centre" == "center").
- The nLab is a category-theory wiki: it usually treats an elementary notion in a more general or
↪  abstract setting. A more general page IS a match when it is genuinely the nLab's primary
↪ treatment of the term ("antiderivative" == "indefinite integral", "sinh" == "hyperbolic sine",
↪  "iff" == "logical equivalence").
- But a page about a DIFFERENT member of the same concept cluster is NOT a match, even though the
↪  nLab redirects one to the other. These are common here and must be rejected: "ceiling" vs "
↪ floor", "imaginary part" vs "real part", "odd integer" vs "even number", "numerator" vs "
↪ fraction", "column operation" vs "row operation".
- A page covering the term as one of several cases may still match if the nLab has no dedicated
↪ page and the term is genuinely treated there ("maximum" vs "extremum" -- judge on whether the
↪ definition's content is actually covered).
- A page whose title merely CONTAINS the term but is about a narrower concept is NOT a match ("
↪ group" vs "group cohomology").
- Beware false friends: the same word names different concepts in different subfields. "bias" (
↪ statistics) is NOT the type-theoretic "biased definition"; "tension" (mechanics) is NOT "Nambu-
↪ Goto action"; "validation" is NOT "judgment".
- When in doubt, prefer null over a wrong match.
Write a JSON array to EXACTLY this path: {outfile}
Each element: {"term": <term>, "match": <chosen nLab page name or null>, "confidence": "high"|"
↪ medium"|"low", "reason": <short phrase>}
Keep every input term, in order. After writing the file, reply with just the count of non-null
↪ matches.
```

Figure 4: Stage-1 judge prompt for the nLab source. Candidate page names are tiered (exact/variant/redirect/phrase) upstream; the rules here specifically guard against the nLab's redirect-driven "concept cluster" failure mode (e.g. "ceiling" redirecting to "floor").

nLab Verification Prompt (Stage 2)

```
You are verifying candidate matches between Oxford Dictionary of Mathematics entries and pages on
↪  the nLab (a category-theory / higher-mathematics wiki), using the ACTUAL nLab page text.
Read this JSON file: {infile}
Each element has "term", "definition" (full Oxford definition), "nlab_page", "nlab_intro" (the
↪ real opening prose of that page), "nlab_stub" (true when the nLab page exists but carries no
↪ definitional prose at all -- typically just a few links), and "prior_confidence"/"source" (
↪ both to be IGNORED -- judge fresh).
For each element decide whether the nLab page, PER ITS INTRO, describes the SAME mathematical
↪ concept as the dictionary definition.
Verdicts:
- "confirmed" -- the same concept. The nLab routinely states an elementary notion in a more
↪ general or abstract setting (a dictionary's "antiderivative" against a page on indefinite
↪ integrals, or a concrete construction stated category-theoretically). That is still "confirmed
↪ " so long as the intro is genuinely about the term's concept.
- "related" -- overlapping or adjacent but NOT the same concept: a sibling in the same cluster (a
↪  "ceiling" definition against a page about the floor function), one case of a broader page, a
↪ specialization, or a page that merely mentions the term.
- "wrong" -- different topic, wrong sense, or a homonym collision across subfields (statistical "
↪ bias" against a type-theoretic "biased definition").
Judge on MEANING, grounded in the intro text, not on title spelling. Singular/plural, spelling
↪ variants and word-order differences are still the same concept.
BIOGRAPHIES: when the Oxford definition is a biographical entry (it opens with life dates, e.g.
↪ "(1802-29) Norwegian mathematician...") and the nLab page is that same person's entry, that is
↪  "confirmed". The nLab's person pages are typically just a list of the person's own writings,
↪ and that list identifies whose page it is -- it is sufficient evidence, short as it is. Use "
↪ related"/"wrong" only if the page is a different person or not a person at all.
Otherwise, if "nlab_stub" is true or the intro is too short to tell, the page carries no content
↪ to check against: use "related" and begin the note with "STUB:" so these stay separable from
↪ genuine concept mismatches. Do NOT infer agreement from the page name alone.
Write a JSON array to EXACTLY this path: {outfile}
Each element: {"term": <term>, "nlab_page": <page>, "verdict": "confirmed"|"related"|"wrong", "
↪ note": <short reason grounded in the intro>}
Keep all input elements, in order. After writing, reply with just a tally "confirmed=X related=Y
↪ wrong=Z".
```

Figure 5: Stage-2 verification prompt for the nLab source, grounded in each page's real fetched prose. Adds a biography carve-out and an explicit stub-detection rule (`STUB:`-prefixed notes) not present in the Wikipedia verification prompt.

Reddit ELI5 Judge Prompt (Single Stage)

```
You are checking whether posts from the subreddit r/explainlikeimfive can serve as plain-language explanations of entries from the Oxford Dictionary of Mathematics.
Read this JSON file: {infile}
Each element has "term", "definition" (the Oxford definition, may be truncated with ...), and "candidates": a list of threads, each with "id", "title" (the question asked), "flair" (topic tag Reddit users assigned) and "answer" (the thread's best answer, truncated).
For each entry pick the ONE candidate thread whose answer explains the same concept as term+definition, or null if none does.
What counts as a match:
- The thread's ANSWER must explain the same concept the definition describes. The question title is context; judge on the answer.
- The answer does not have to be rigorous or complete -- it is a lay explanation, and being informal is not a reason to reject it. It has to be ABOUT the same thing.
- A thread asking about one narrow aspect, application, or consequence of the term is NOT a match unless its answer still explains what the term is (e.g. "the abacus finger theory" explains a bead technique, not what an abacus is -> null).
- Beware word-sense collisions, which are the main failure mode here: the same word names different things in different fields, and this subreddit is general-audience. "matrix" (the film), "argument" (a disagreement), "acre" (real estate), "aggregate" (economics) are not the mathematical senses. Use the definition to disambiguate.
- A thread comparing the term with something else counts if the answer defines the term along the way (e.g. "difference between a t-test and ANOVA" does explain ANOVA). - If several threads qualify, pick the one whose answer explains the concept most directly and completely.
- The definition may start with "See SOMETHING. --"; the text after the dash is the resolved cross-reference and is the real definition.
- When in doubt prefer null. A wrong match is worse than a miss. Write a JSON array to EXACTLY this path: {outfile}
Each element: {"term": <term>, "match": <chosen candidate "id" string, or null>, "confidence": "high"|"medium"|"low", "reason": <short phrase>}
IMPORTANT: "match" must be the candidate's "id" field, not its title.
Keep every input term, in order. After writing the file, reply with just the count of non-null matches.
```

Figure 6: Single-stage judge prompt for r/explainlikeimfive. Unlike the wiki sources, the judge here already conditions on real content (the thread's fetched answer text, not just its title), so no separate verification stage follows; the output confidence label (high/medium/low) is emitted directly rather than a confirmed/related/wrong verdict.

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

Soumyadeep Nag[1]* **and Suparnojit Sarkar**[2]
[1]Indian Statistical Institute, Kolkata, India
[2]Heritage 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.

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 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.

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. Contact: soumyadeepnag2@gmail.com

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 305–311
October 29, 2026 ©2026 Association for Computational Linguistics

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

$$K(x) = \mathop{\Big\|}_{k \in \mathcal{D}:\, k \subseteq x} \mathcal{D}[k], \tag{1}$$

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

$$\begin{aligned} r_i &= \frac{\exp(s_{i,\text{YES}})}{\exp(s_{i,\text{YES}}) + \exp(s_{i,\text{NO}})}, \\ z_i &= \mathbb{1}(r_i \geq \gamma), \end{aligned} \tag{2}$$

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

$$z_i = \mathbb{1}(\texttt{"Yes"} \in g_i), \tag{3}$$

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

$$P^\star = \begin{cases} P_{\text{sel}}, & |P_{\text{sel}}| > 0, \\ P, & |P_{\text{sel}}| = 0. \end{cases} \tag{4}$$

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 $P^\star$, H, and $K(x)$. For generated tokens $y_{1:L}$ and next-token logits a_t, the implementation computes

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

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

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 $P^\star$. With $\tau = 0.72$, the actual route is

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

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 B1–B4.

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.

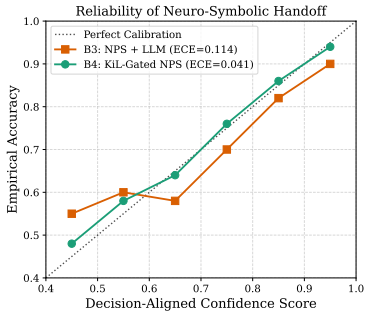
Diagnostic	Corrected-JSON audit
Shortcut	96.1 macro-F1; 89.7% of label entropy; target-group held out
Reuse	4 hypotheses, 50 targets, and 320 repeated configurations
Provenance	2 invariant distractors; no human-labeled or naturally sourced items

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.

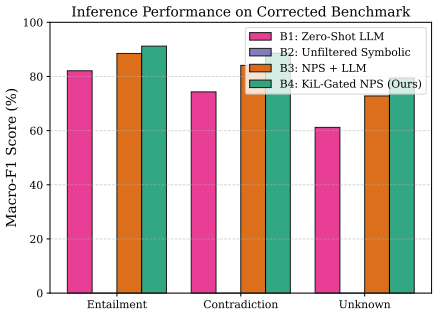
Hypothesis-only diagnostic. The target-group-held-out hypothesis baseline obtains 96.0% accuracy and 96.1 macro-F1 on the corrected JSON. Only 20 divisible-by-five entailments are missed; the other three hypothesis strings determine their labels. The literal hypothesis carries 1.375 of the label's 1.533 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 codebase, 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 $\mathcal{H}(Y \mid H) = 0.159$ 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 (and (<= y 1) (<= z x)) (distinct y
      1) (distinct z x)) (not (and (= y * z x)))))

(declare-const c_int (Int 7
````

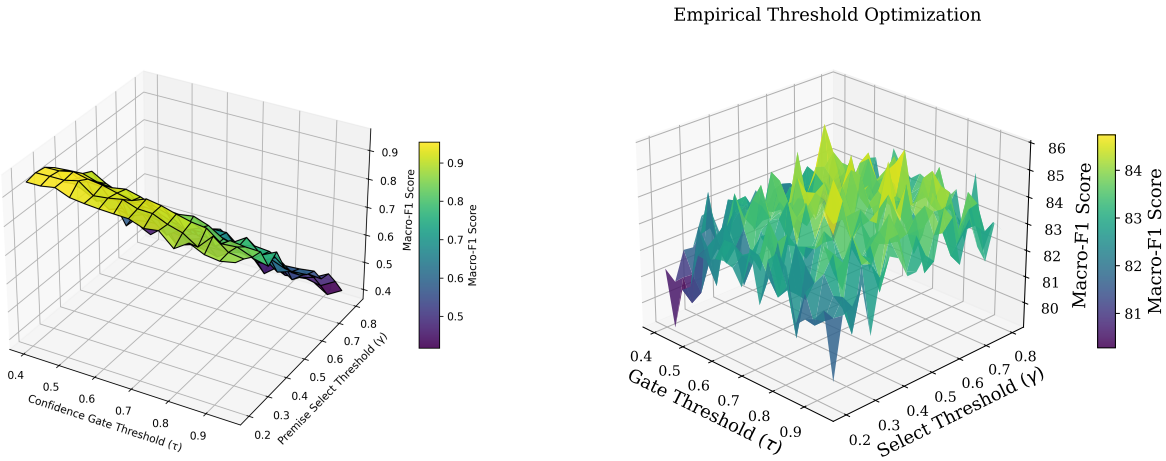
Thus $c = 0.88$ is high sequence likelihood for a string that cannot be executed by the recorded Python wrapper; it is not evidence of formal correctness.

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 evaluation 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.



(a) Surface without sweep records

(b) Analytic noisy surface

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.

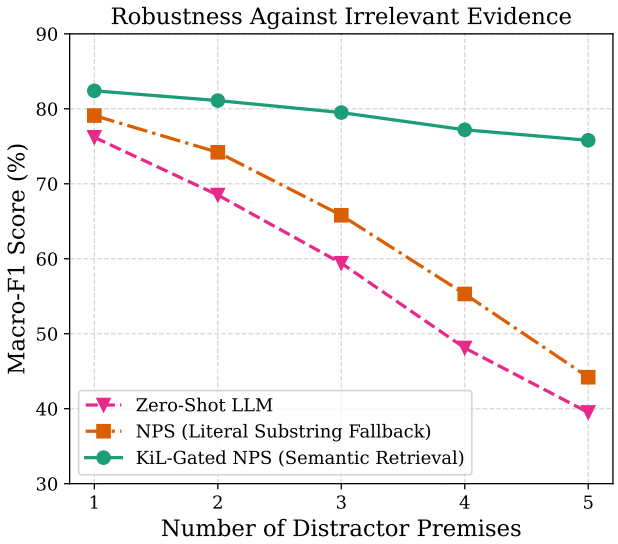


Figure 3: Distractor-count curve retained for provenance. The plotted series are not accompanied by stored per-example predictions or an executed distractor sweep and are excluded from quantitative comparisons.

Proof Identity and Divisible Correspondence in Multi-Derivation Reasoning Evaluation

Toprak Seda Karaosmanoğlu
Middle East Technical University
Ankara, Türkiye
toprak.karaosmanoglu@metu.edu.tr

Abstract

Systems that return several valid derivations require explicit choices about proof identity and proof-set correspondence. This study examines both choices on ProofWriter. An application-aware representation distinguishes grounded uses of the same source rule through canonical premise derivations. Source-ID projection collapses distinct reference proofs in 28 of 400 stratified derivable questions (7.0%). The comparison includes exact proof-set F_1, flattened-atom F_1, Optimal Subpattern Assignment with Jaccard distance (Jaccard-OSPA), and MCT-J. The label MCT-J denotes a balanced optimal-transport distance with Jaccard distance as the ground metric. On 160 fixed outputs from 32 derivable questions, source-ID projection lowers MCT-J distance by 0.0606 on average. A cluster bootstrap over the underlying ProofWriter worlds gives a 95% interval of 0.0316–0.0955. MCT-J equals Jaccard-OSPA on all 117 equal-cardinality outputs, while Jaccard-OSPA has the larger distance on 38 of 43 unequal-cardinality outputs. Of 433 generated proof members, 430 pass independent rule-level replay. Excluding the three replay failures changes the mean representation effect from 0.0606 to 0.0589. Proof identity and proof-set correspondence therefore change structural evaluation on fixed outputs.

1 Introduction

Proof-producing systems expose derivations together with final answers. PRover and ProofWriter generate proof graphs (Saha et al., 2020; Tafjord et al., 2021), and multiPRover generates several alternative proofs for one answer (Saha et al., 2021). Structured evaluation must therefore specify when two derivations count as the same proof and how reference and predicted proof sets are compared.

Proof identity matters when one source rule is applied more than once. Two grounded applications can cite the same rule statement and depend on different premise derivations. A representation based on the source identifier (source ID) maps those applications to the same rule node. This changes the represented proof before a proof-set distance is computed. The application-aware representation keeps the applications distinct through canonical premise derivations.

Proof-set correspondence matters when several proofs are present. Exact recovery scores shared proof members. Optimal Subpattern Assignment (OSPA) uses graded one-to-one matching with a cardinality penalty (Schuhmacher et al., 2008). Balanced optimal transport permits normalized proof mass to divide across several counterparts (Peyré and Cuturi, 2019). The paper uses the label MCT-J for the resulting balanced optimal-transport distance with Jaccard distance as the ground metric. One-to-one and divisible correspondence agree at equal cardinality and can separate when cardinalities differ.

ProofWriter offers a controlled setting because provable examples contain the complete proof set under the generated Datalog semantics (Tafjord et al., 2021). Under the open-world assumption (OWA), the depth-5 test split contains 11,376 derivable questions, including 2,456 with multiple reference proofs. The stratified reference analysis finds source-ID proof-set collapse in 28 of 400 derivable questions. On 160 fixed derivable outputs from 32 questions, the mean MCT-J difference between application-aware and source-ID scoring is 0.0606. A cluster bootstrap over the underlying ProofWriter worlds gives a 95% interval of 0.0316–0.0955.

The paper develops a canonical grounded-application representation and tests its consequences in a stratified reference analysis that distinguishes repeated source-rule use from observed proof-set collapse. It then formalizes divisible proof-set correspondence against one-to-one Jaccard-OSPA and measures the effect by rescoring the same outputs under both representations. Deterministic rule replay and sensitivity analyses

The Proceedings of the 4th Workshop on Mathematical Natural Language Processing (MathNLP 2026), pages 312–321
October 29, 2026 ©2026 Association for Computational Linguistics

test the result across replay filtering and the stated encoding choices.

2 Related Work

Proof generation and reasoning-chain evaluation. RuleTaker studies reasoning over natural-language facts and rules (Clark et al., 2020). PRover adds proof generation (Saha et al., 2020), ProofWriter supplies generated implications with proofs (Tafjord et al., 2021), and multiPRover generates multiple proof graphs (Saha et al., 2021). EntailmentBank represents multistep explanations as entailment trees (Dalvi et al., 2021). Recent studies evaluate reasoning steps and reasoning-chain verifiers (Prasad et al., 2023; Jacovi et al., 2024). Other work studies proof-tree generation (Gupta, 2026) and formal logical reasoning (López-Ponce and Bel-Enguix, 2025). Adjacent settings include tool-integrated mathematical reasoning (Obozov et al., 2025) and autoformalization evaluation (Poiroux et al., 2025). The present study focuses on evaluation when one answer has several explicit derivations.

Structured and evidence-set evaluation. Structured metrics can compare outputs through matching of common substructures (Chen et al., 2023). Graph evaluation includes many-to-many alignment (Opitz et al., 2021). Fact-checking research also evaluates retrieved evidence and evidence quality (Thorne et al., 2018; Schlichtkrull et al., 2024; Akhtar et al., 2026). The evaluated object is a set of derivation proofs whose internal rule applications also require an identity convention.

Finite-set assignment and transport. Optimal Subpattern Assignment compares finite sets by one-to-one assignment and a cardinality term (Schuhmacher et al., 2008). Optimal transport compares discrete measures through marginal-constrained couplings (Peyré and Cuturi, 2019). Structured optimal transport extends this framework to structured data (Alvarez-Melis et al., 2018). Jaccard distance is fixed as the proof-level ground cost so that Jaccard-OSPA and MCT-J differ through their cross-set correspondence.

3 Task and Proof Representation

3.1 Benchmark object

A ProofWriter instance contains natural-language facts and rules, a question, a truth status in $\{\text{true}, \text{false}, \text{unknown}\}$, and the complete proof set for each provable answer under its generated Datalog theory (Tafjord et al., 2021). A proof is a rooted derivation directed acyclic graph (DAG) whose fact steps feed rule-application steps and whose root supports the queried status.

For a derivable question q, let

$$G_q = \{g_1, \ldots, g_m\}, \qquad P_q = \{p_1, \ldots, p_n\}$$

denote the sets of distinct reference and predicted proofs. Exact duplicate predicted proofs collapse to one member before proof-set scoring.

3.2 Grounded application identity

Source identity records the rule statement used by a derivation step. Grounded application identity incorporates the premise derivations into the step identity. Canonical identities are defined recursively. A fact step with source fact f receives

$$\sigma(\text{fact}) = (\text{fact}, f). \tag{1}$$

For an application of source rule r to canonical premise derivations $\sigma_1, \ldots, \sigma_k$,

$$\sigma(\text{apply}) = (\text{apply}, r, \text{sort}(\sigma_1, \ldots, \sigma_k)). \tag{2}$$

ProofWriter rules use conjunction-style antecedents, so the canonical premise collection is order-invariant. The paper refers to this construction as the *application-aware representation*. Distinct grounded applications of one source rule receive distinct identities whenever their canonical premise derivations differ.

Proposition 1 (Step-name invariance)**.** *Renaming local step identifiers while preserving the rooted derivation DAG, source IDs, and premise relations leaves the application-aware canonical proof unchanged.*

Proof. Consider a topological ordering of the DAG. A fact signature depends only on its source fact. For a rule-application step, the induction hypothesis fixes every premise signature. Eq. (2) then depends only on the source rule and the resulting canonical premise signatures. The root signature and all structural atoms derived from these signatures remain unchanged.

3.3 Source-ID projection

The source-ID representation identifies facts by source fact ID and rule applications by source rule ID. Edges connect these source-level nodes. Repeated grounded applications can therefore map to

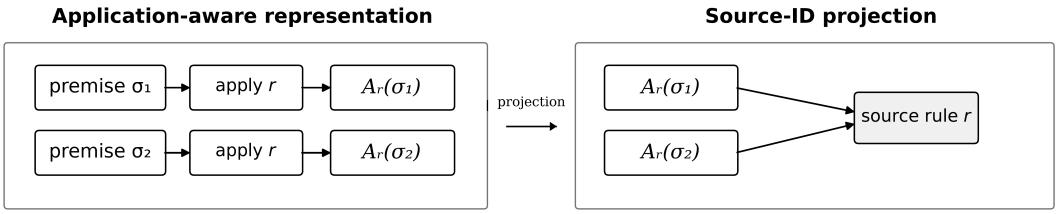


Figure 1: Grounded rule-application identity and source-ID projection. The application-aware representation distinguishes applications of source rule r through their canonical premise derivations σ_1 and σ_2. Here $A_r(\sigma)$ denotes the grounded application identity of source rule r with canonical premise derivation σ. Source-ID projection maps both application identities to the same source-rule node.

one rule node, as illustrated in Figure 1. Generation uses the application-aware output structure. Source-ID projection is applied later when the same generated DAGs are rescored.

The stratified reference analysis contains direct many-to-one projection cases. One question has four distinct application-aware reference proofs and three distinct source-ID proofs after projection and deduplication. Section 6.1 reports the prevalence of this behavior.

4 Comparing Alternative Proof Sets

The application-aware encoding uses node atoms for canonical step signatures, source-label atoms for source IDs, directed premise-to-application edge atoms, and one root atom. The primary source-ID encoding uses source-ID node atoms, directed edges, and one root atom. This design omits separate source-label atoms from the source-ID representation. Fact identities are already carried by fact-node identifiers in both encodings. Section 6.5 tests a matched atom inventory by adding source-label atoms to the source-ID representation. Table 1 summarizes the four structural measures.

4.1 Exact and flattened comparators

Exact proof-set F_1 treats G and P as sets of canonical proofs:

$$F_1^{\text{exact}}(G,P) = \frac{2|G \cap P|}{|G| + |P|}. \tag{3}$$

When both proof sets are empty, exact F_1 is 1. When exactly one proof set is empty, exact F_1 is 0. Flattened-atom F_1 first unions structural atoms across all proofs and then computes set F_1. This comparator measures structural overlap after proof grouping has been removed. Exact duplicates collapse under the same set policy before all proof-set measures are computed.

4.2 Jaccard ground distance and OSPA

For proof atom sets g and p,

$$d_J(g,p) = 1 - \frac{|g \cap p|}{|g \cup p|}. \tag{4}$$

Every represented proof contains a root atom, so the atom sets used by d_J are nonempty. OSPA is instantiated with order parameter 1, cutoff $c = 1$, and d_J as the ground distance (Schuhmacher et al., 2008). Since $d_J \leq 1$, the cutoff is inactive. For $m \leq n$,

$$d_{\text{OSPA}}(G,P) = \frac{1}{n}\left[\min_{\tau \in \mathcal{I}_{m,n}} \sum_{i=1}^{m} d_J\big(g_i, p_{\tau(i)}\big) + (n-m)\right]. \tag{5}$$

where $\mathcal{I}_{m,n}$ is the set of injections from $\{1,\ldots,m\}$ into $\{1,\ldots,n\}$. The case $m > n$ uses the symmetric expression. This study refers to the resulting instantiation as Jaccard-OSPA.

4.3 Balanced transport over proof sets

For proof sets with at least one member, uniform empirical measures assign equal mass to each distinct proof:

$$\mu_G = \frac{1}{m}\sum_{i=1}^{m} \delta_{g_i}, \qquad \mu_P = \frac{1}{n}\sum_{j=1}^{n} \delta_{p_j}. \tag{6}$$

Let $\mathbf{a} = (1/m,\ldots,1/m)$ and $\mathbf{b} = (1/n,\ldots,1/n)$. The balanced transport polytope is

$$\Pi(\mathbf{a},\mathbf{b}) = \left\{ \pi \in \mathbb{R}_+^{m\times n} \;\middle|\; \begin{matrix} \pi \mathbf{1}_n = \mathbf{a}, \\ \pi^\top \mathbf{1}_m = \mathbf{b} \end{matrix} \right\}. \tag{7}$$

The transport distance is

$$d_{\text{MCT}-\text{J}}(G,P) = \min_{\pi \in \Pi(\mathbf{a},\mathbf{b})} \sum_{i=1}^{m}\sum_{j=1}^{n} \pi_{ij} d_J(g_i,p_j). \tag{8}$$

Measure	Proof-level comparison	Cross-set correspondence	Cardinality treatment	Role in the analysis
Exact proof-set F_1	Exact canonical proof identity	Exact member overlap	Set sizes enter the F_1 denominator	Strict proof-set recovery
Flattened-atom F_1	Exact structural-atom identity	Union of atoms across proofs	Proof grouping and proof-set cardinality are removed by flattening	Control for loss of proof grouping
Jaccard-OSPA	Jaccard distance between proofs	Optimal one-to-one assignment	Unmatched-member penalty with order 1, cutoff 1	Graded finite-set comparator
MCT-J	Jaccard distance between proofs	Balanced divisible coupling	Uniform marginal mass over all reference and predicted proofs	Graded comparison with divisible matching

Table 1: Structural measures used in the study. Jaccard-OSPA and MCT-J share the same proof-level ground distance and differ in cross-set correspondence.

The label MCT-J denotes this balanced optimal-transport distance with Jaccard distance as the ground metric. It is the standard Wasserstein-1 construction (Peyré and Cuturi, 2019). When both proof sets are empty, the distance is 0. When exactly one proof set is empty, the distance is 1. After duplicate collapse, MCT-J is a metric on finite represented proof sets. Jaccard distance is a metric on finite atom sets, and Wasserstein-1 inherits metricity from its ground distance. Duplicate collapse makes the represented proof-set map injective. Nonempty-set distances are bounded by 1, so the empty-set extension preserves the triangle inequality. The resulting distance is permutation-invariant and bounded in $[0, 1]$. Uniform weighting assigns mass $1/m$ to each reference proof and $1/n$ to each predicted proof. Proof-set size therefore enters the transport geometry through the marginals. ProofWriter leaves proof importance unspecified. The primary analysis uses uniform proof mass, and Section 6.5 tests size-weighted mass.

4.4 Relation to one-to-one assignment

The common Jaccard ground distance isolates the cross-set correspondence difference between Jaccard-OSPA and MCT-J.

Proposition 2 (Equal-cardinality equivalence)**.** *For $|G| = |P| = k > 0$ with uniform proof weights,*

$$d_{\mathrm{MCT-J}}(G, P) = d_{\mathrm{OSPA}}(G, P).$$

Proof. Multiplying every feasible transport entry by k yields a doubly stochastic matrix. A linear objective attains an optimum at an extreme point of this polytope. The Birkhoff–von Neumann theorem identifies these extreme points with permutation matrices. Scaling by $1/k$ yields the minimum average one-to-one Jaccard assignment cost. Jaccard-OSPA has the same objective at equal cardinality because its cardinality term equals zero.

Proposition 3 (Jaccard-OSPA upper bound)**.** *For finite proof sets with at least one member under the definitions above,*

$$d_{\mathrm{MCT-J}}(G, P) \leq d_{\mathrm{OSPA}}(G, P).$$

Proof. Assume $m \leq n$. Symmetry gives the corresponding construction for $m > n$. Take an OSPA-optimal injection from the m reference proofs to m predicted proofs and send mass $1/n$ along every assigned pair. Each reference row retains mass $1/m - 1/n$, with total residual mass $(n - m)/n$. Route this residual mass to the $n - m$ unmatched prediction columns while satisfying the uniform marginals. Since $d_J \leq 1$, residual transport costs at most $(n - m)/n$. Assigned transport costs $(1/n) \sum_i d_J(g_i, p_{\tau(i)})$. Their sum is bounded by the OSPA objective, and the optimal balanced transport cost is bounded by this feasible plan.

Proposition 3 makes unequal proof-set cardinality a requirement for a positive OSPA-minus-MCT-J gap. Some unequal-cardinality pairs still yield equality. A merge example illustrates the separation: if G contains k distinct singleton proof-atom sets and P contains their union as one proof, then

$$d_{\mathrm{MCT-J}} = 1 - \frac{1}{k}, \quad d_{\mathrm{OSPA}} = 1 - \frac{1}{k^2}.$$

The difference $1/k - 1/k^2$ arises from correspondence and cardinality treatment under the same Jaccard ground distance.

5 Data and Evaluation

5.1 Reference data and stratified reference analysis

The study uses the depth-5 test split of ProofWriter OWA V2020.12.3. The test split contains 20,030 questions: 5,688 true, 5,688 false, and 8,654 unknown. Among 11,376 derivable questions, 2,456 have at least two reference proofs (21.59%).

Source-ID representation loss is analyzed on 400 stratified derivable test questions: 80 with one reference proof, 120 with two, 120 with three or four, and 80 with five to eight. These rates describe the stratified reference analysis, while full-test counts provide corpus prevalence.

A high-multiplicity evaluation set extends the stratified reference analysis beyond eight proofs. It contains 40 questions with 17–32 reference proofs and mean proof-set size 21.48. The set is deliberately enriched for questions with large reference proof sets.

5.2 Generation sample

The generation sample contains 40 questions, with eight questions from each proof-set-size stratum 0, 1, 2, 3–4, and 5–8. The 400-question stratified reference sample, the 40-question generation sample, and the 40-question high-multiplicity set were selected deterministically before model generation and are pairwise disjoint. The 32 derivable generation questions span 30 source worlds, where a source world is an underlying generated fact-and-rule theory shared by one or more questions. Five presentation orders per question produce 200 outputs in total, of which 160 correspond to derivable questions. Each presentation deterministically permutes the supplied facts and rules while preserving source IDs and natural-language content.

The public input contains the facts and rules, the question, neutral source IDs, and task instructions. Reference answers, proof annotations, and canonical signatures are used only for evaluation. The task instruction allows at most eight returned derivations in every generation call. Every derivable question in the generation sample has at most eight reference proofs. Returned proofs use a rooted DAG encoded in JavaScript Object Notation (JSON). Generation was run on 8 August 2026 through OpenAI using GPT-5.6 Sol (OpenAI, 2026). Requests used the model identifier `gpt-5.6-sol` with medium reasoning effort. A strict JSON schema and an 8,192-token output budget constrained generation.

Exact proof-set recovery is reported as a descriptive comparator. The primary comparison is the score change produced by applying two representations and two correspondence rules to the same fixed outputs.

Generated DAGs are checked for valid source IDs, step references, roots, cycles, and orphan steps, meaning steps disconnected from the returned root. Exact canonical matches identify ProofWriter reference proofs. Other structurally valid DAGs receive graded structural distances. A separate deterministic rule-replay procedure executes cited rules from the literals established by their premise steps and checks the supported root.

5.3 Paired comparison and uncertainty

Representation effects are paired at the generated-output level. The primary score difference is the mean difference between application-aware and source-ID scoring across 160 derivable outputs. For uncertainty, the 30 source worlds are sampled with replacement as clusters. Each sampled world contributes all associated questions and presentations, preserving within-world dependence and observed cluster sizes. The paired mean is recomputed in 100,000 replicates with seed 20260808, and percentile 95% intervals are reported. These intervals summarize variation across source worlds. Proof-set-size analyses retain the four derivable strata.

Sensitivity settings. For atom-weighted analyses, weighted Jaccard distance is $d_{J,w}(A, B) = 1 - \sum_{a \in A \cap B} w(a) / \sum_{a \in A \cup B} w(a)$. Node and root weights remain 1, while source-label and edge weights take the tested values. For proof-mass analyses, each proof receives normalized mass proportional to its atom count raised to α. Proposition 2 applies to the uniform-mass case $\alpha = 0$.

6 Results

6.1 Stratified reference analysis

Across all 20,030 test questions, 2,928 (14.62%) contain at least one reference derivation that applies a source rule more than once. Conditioned on the 11,376 derivable questions, the same count is 25.74%. The stratified reference analysis contains 82 of 400 questions (20.5%) with such within-proof reuse. Source-ID projection reduces the number of distinct reference proofs in 28 of 400 questions (7.0%), removing 37 reference proof members through deduplication. All 28 collapse cases

occur within the 82-question subset with repeated source-rule use. By proof-set-size stratum, collapse occurs in 0 of 80 one-proof questions, 0 of 120 two-proof questions, 16 of 120 questions with three or four proofs, and 12 of 80 questions with five to eight proofs. In the separate 17–32-proof high-multiplicity evaluation set, source-ID projection collapses distinct reference proofs in 32 of 40 questions and removes 130 of 859 application-aware reference proofs (15.13%). Table 2 collects the principal reference-data quantities.

Repeated source-rule use records reuse of one source rule inside a derivation. Proof-set collapse records the loss of distinct proof members after source-ID projection. Their counts therefore answer different questions. Collapse appears only in the larger proof-set strata of the stratified reference analysis, which motivates the paired rescoring analysis by proof-set size.

6.2 Output integrity and proof-set cardinality

Across 200 generated outputs, including the eight unknown questions under five presentations, truth-status accuracy and JSON parseability are both 100%. All 433 generated proof members in the 160 derivable outputs pass the structural DAG checks. Application-aware deduplication retains all 433 members, while source-ID projection merges one predicted pair in one output. Twelve of the 160 derivable outputs reach the eight-proof output limit. The mean predicted proof-set size is 2.71 and the mean reference proof-set size is 3.06. Predicted and reference cardinalities match for 117 of 160 outputs (73.1%). Predicted proof sets are smaller in 27 of 160 outputs (16.9%) and larger in 16 of 160 outputs (10.0%).

6.3 Representation effect on fixed outputs

Table 3 reports application-aware and source-ID scores for the same 160 derivable outputs. Application-aware MCT-J averages 0.1447 and source-ID MCT-J averages 0.0842. The paired difference between application-aware and source-ID scoring is 0.0606, with a 95% cluster bootstrap interval over the 30 source worlds of 0.0316–0.0955. Forty-one outputs change, and every changed output receives a lower MCT-J distance under source-ID projection. Relative to the application-aware mean, source-ID projection reduces the measured distance by 41.8% in this paired rescoring analysis.

The five presentation orders have positive mean MCT-J differences ranging from 0.0413 to 0.0880.

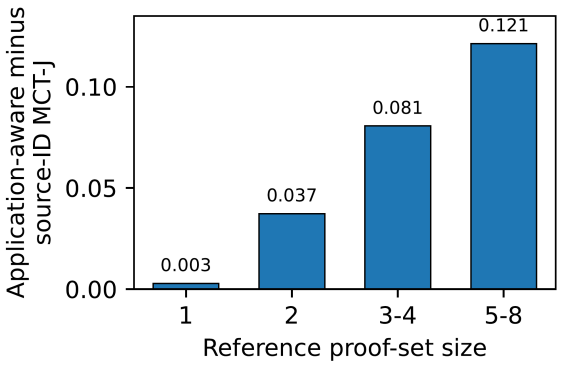


Figure 2: Mean MCT-J difference between application-aware and source-ID scoring by reference proof-set size. Positive values correspond to larger distance under application-aware scoring.

Score changes also extend beyond proof-set collapse. Within the 32-question generation sample, source-ID projection changes reference proof-set size for one question and predicted proof-set size for one generated output. For most of the 41 MCT-J changes, projection leaves both represented proof-set sizes unchanged, so the score change comes from altered grounded-application structure within proof members.

The mean representation effect rises with reference proof-set size (Figure 2): 0.0028 for one proof, 0.0373 for two proofs, 0.0807 for three or four proofs, and 0.1214 for five to eight proofs. Across 32 questions, the descriptive Spearman association between reference proof-set size and the item-level mean representation effect is $\rho = 0.665$. The association pertains to the paired rescoring analysis.

Exact proof-set F_1 changes on 6/160 outputs. Flattened-atom F_1 changes on 44/160 outputs. Its mean application-aware minus source-ID difference is -0.0641, with a 95% cluster bootstrap interval over the 30 source worlds of $[-0.0980, -0.0349]$. Source-ID projection therefore changes exact proof identity and the atom geometry seen by graded measures.

6.4 Divisible correspondence and cardinality mismatch

For the 117 application-aware outputs with equal predicted and reference proof-set sizes, MCT-J and Jaccard-OSPA have identical values in every case. Among the 43 unequal-cardinality outputs, Jaccard-OSPA has the larger distance in 38 cases and the two distances are equal in five. The

Analysis quantity	Count	Rate
Multi-proof among derivable test questions	2,456/11,376	21.59%
Repeated source-rule use, all test questions	2,928/20,030	14.62%
Repeated source-rule use, stratified reference analysis	82/400	20.50%
Source-ID proof-set collapse, stratified reference analysis	28/400	7.00%
Reference proof members removed by projection	37	–

Table 2: Reference-data quantities. The 2,928/20,030 rate uses all test questions to report corpus prevalence. The stratified-reference rates use the sampled derivable strata.

Measure	Application-aware	Source-ID	Δ	Changed
Exact proof-set F_1 ↑	.8657	.8697	-.0041	6/160
Flattened-atom F_1 ↑	.8744	.9384	-.0641	44/160
MCT-J ↓	.1447	.0842	+.0606	41/160
Jaccard-OSPA ↓	.1713	.1661	+.0052	8/160

Table 3: Paired rescoring of the same outputs. Δ denotes application-aware minus source-ID. Arrows mark the favorable direction of each measure.

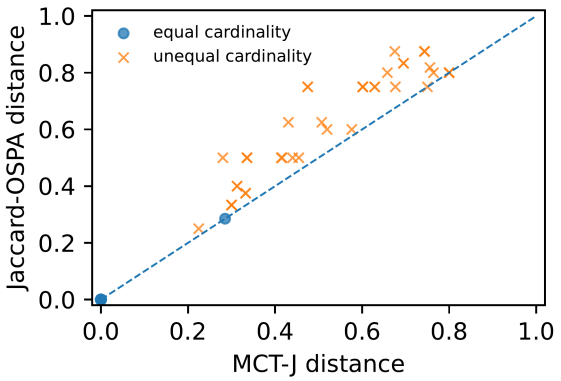


Figure 3: Application-aware MCT-J and Jaccard-OSPA on 160 outputs. Equal-cardinality outputs lie on the diagonal. Unequal-cardinality outputs may separate because balanced transport permits divisible correspondence.

mean OSPA-minus-MCT-J difference over the 43 unequal-cardinality outputs is 0.0988. Figure 3 displays both regimes.

Equal cardinality reproduces the one-to-one optimum established in Proposition 2. Reference and predicted proof-set sizes differ in 43 of 160 generated derivable outputs (26.9%). The eight outputs with a Jaccard-OSPA change are a subset of the 41 outputs with a MCT-J change. This leaves 33 outputs where only MCT-J changes, and reference and predicted proof-set sizes differ in all 33. In each case the Jaccard-OSPA value equals its cardinality penalty because its assigned proof pairs have zero Jaccard cost. Balanced transport still redistributes mass across the unequal proof sets, so MCT-J remains sensitive to the representation change.

6.5 Rule replay and sensitivity analyses

The rule-replay procedure first reproduces all 2,210 reference proof roots from the stratified reference, generation, and high-multiplicity evaluation sets. These roots contain 4,085 fact nodes and 9,659 rule applications, with zero replay failures. Of 433 generated proof members, 430 pass independent rule-level replay (99.31%). All returned proof members pass replay for 159 of 160 derivable outputs. The three failures occur in one presentation of one question, where the listed premise fails to establish the cited rule antecedent. Excluding these three proof members yields application-aware MCT-J 0.1420 and source-ID MCT-J 0.0831. The paired representation effect is 0.0589 with a 95% cluster bootstrap interval of 0.0297–0.0940, compared with the primary 0.0606 [0.0316, 0.0955]. Renaming local step identifiers and reordering steps and premises leaves the application-aware atom set unchanged for all 433 generated proof members.

One sensitivity analysis adds source-label atoms to the source-ID representation so that both representations use the same structural atom types. The resulting effect is 0.0634 [0.0334, 0.0996], with the same 41 changed outputs. Recursive signature depth is evaluated at depth 1, depth 2, and full recursion. Source-label and edge-atom weights take values in $\{0.5, 1, 2\}$. A one-at-a-time ablation removes each atom type in turn. Proof mass uses $\alpha \in \{0, 0.5, 1\}$ under the definition in Section 5.3. Across these analyses, the effect ranges from 0.0306 to 0.0723, and every reported pointwise 95% cluster bootstrap interval remains above zero.

The adversarial perturbation analysis uses the 24 derivable generation questions with at least two reference proofs. For each question, one reference proof is omitted in turn to form an omission-only predicted family. A different retained proof

serves as the donor. Deleting each non-root donor atom once creates the candidate near-duplicates. Each distinct candidate is added to the predicted family and scored against the original reference family. The construction yields 6,032 perturbations and tests whether a structurally redundant predicted alternative can improve the score. Executable derivation status lies outside this structural diagnostic. Across the perturbations, MCT-J distance decreases in 2,575 cases, increases in 3,032, and ties in 425. Using perturbed minus baseline distance, the mean signed change is -0.000015. Jaccard-OSPA distance decreases in 5,382 cases and ties in 650, with mean decrease 0.04017.

7 Discussion

The stratified reference analysis isolates a representation effect. Repeated source-rule use is more frequent than proof-set collapse, yet source-ID projection removes distinct reference proofs in 28 of 400 stratified questions. On fixed model outputs, projection also changes grounded-application atoms while the projection usually leaves the represented proof-set sizes unchanged. This explains why MCT-J changes on 41 outputs even though projection changes predicted proof-set size once.

Proof-set size localizes the effect. The one-proof stratum has a mean difference of 0.0028, rising to 0.1214 for five-to-eight-proof questions. Equal-cardinality proof sets reproduce the same one-to-one optimum under Jaccard-OSPA and MCT-J. Unequal cardinality permits divisible transport and produces the observed gap. Across the same fixed outputs, source-ID scoring raises both F_1 measures and lowers both distance measures. All four measures therefore place the source-ID representation closer to the reference on average.

Rule replay changes the mean representation effect from 0.0606 to 0.0589 after the three replay failures are excluded. The sensitivity checks preserve its direction across the tested representation and weighting choices. Under the structural near-duplicate perturbation, Jaccard-OSPA decreases in 5,382 of 6,032 cases with mean decrease 0.04017, while the mean signed MCT-J change is near zero. The two measures therefore respond differently to structurally near-duplicate predicted members.

8 Conclusion

Multi-derivation evaluation requires explicit choices about proof identity and proof-set correspondence. Grounded application identity preserves distinctions between repeated grounded uses of one source rule. Source-ID projection can collapse those distinctions. In the stratified reference analysis, source-ID projection collapses distinct reference proofs in 28 of 400 derivable questions. On the same 160 generated outputs, it lowers MCT-J distance by 0.0606 on average. Rule replay and sensitivity analyses preserve the direction of this effect. The 17–32-proof high-multiplicity evaluation set extends the collapse result to larger reference proof sets. MCT-J equals Jaccard-OSPA at equal cardinality. Under cardinality mismatch, Jaccard-OSPA has the larger distance for 38 of 43 outputs. Representation and correspondence therefore change structural evaluation on fixed outputs. Explicit specification of both choices supports reproducible evaluation of alternative derivations.

Limitations

ProofWriter uses synthetic rule reasoning under generated Datalog semantics. Transfer to natural mathematical proofs remains a future empirical setting. The primary paired rescoring analysis uses 32 derivable questions from GPT-5.6 Sol. The main experiment evaluates one model, so cross-system evidence remains limited. The stratified reference analysis examines 400 derivable questions. The bootstrap summarizes variation across 30 source worlds, so interval precision is tied to that cluster count.

The structural atom types define the Jaccard geometry. Other structural encodings can produce different local Jaccard distances. The reported sensitivity analyses cover the tested recursion, atom types, weighting, and proof-mass choices. Exact duplicates collapse under set semantics. Structurally distinct near-duplicates retain separate mass. The adversarial perturbation analysis shows that some can lower MCT-J. Human proof-quality judgments and robustness to deliberately constructed near-duplicate proof sets remain separate validation targets.

The output format allows at most eight predicted proofs. Every derivable question in the generation sample has at most eight reference proofs, so its complete reference set fits within this limit. The 17–32-proof high-multiplicity evaluation set extends the analysis of reference proof sets beyond this range. The eight-proof output limit restricts anal-

ysis of larger predicted proof sets. ProofWriter is public, so training-data exposure is possible for contemporary systems.

Evaluation on less controlled natural-language reasoning and on other datasets requires separate studies. The conclusions concern structural evaluation.

References

Mubashara Akhtar, Michael Schlichtkrull, and Andreas Vlachos. 2026. Ev2R: Evaluating evidence retrieval in automated fact-checking. *Transactions of the Association for Computational Linguistics*, 14:530–561.

David Alvarez-Melis, Tommi Jaakkola, and Stefanie Jegelka. 2018. Structured optimal transport. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1771–1780. PMLR.

Yunmo Chen, William Gantt, Tongfei Chen, Aaron Steven White, and Benjamin Van Durme. 2023. A unified view of evaluation metrics for structured prediction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12868–12882, Singapore. Association for Computational Linguistics.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3882–3890. International Joint Conferences on Artificial Intelligence Organization.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Aayushee Gupta. 2026. From sentences to proof trees: Leveraging language models for structured reasoning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 967–981, Rabat, Morocco. Association for Computational Linguistics.

Alon Jacovi, Yonatan Bitton, Bernd Bohnet, Jonathan Herzig, Or Honovich, Michael Tseng, Michael Collins, Roee Aharoni, and Mor Geva. 2024. A chain-of-thought is as strong as its weakest link: A benchmark for verifiers of reasoning chains. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4615–4634, Bangkok, Thailand. Association for Computational Linguistics.

Francisco F. López-Ponce and Gemma Bel-Enguix. 2025. Into the limits of logic: Alignment methods for formal logical reasoning. In *Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025)*, pages 112–123, Suzhou, China. Association for Computational Linguistics.

Mark Obozov, Michael Diskin, Aleksandr Beznosikov, Alexander Gasnikov, and Serguei Barannikov. 2025. Synthetic proofs with tool-integrated reasoning: Contrastive alignment for LLM mathematics with Lean. In *Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025)*, pages 195–202, Suzhou, China. Association for Computational Linguistics.

OpenAI. 2026. GPT-5.6 Sol model. OpenAI API documentation. Accessed 19 September 2026.

Juri Opitz, Angel Daza, and Anette Frank. 2021. Weisfeiler-Leman in the Bamboo: Novel AMR graph metrics and a benchmark for AMR graph similarity. *Transactions of the Association for Computational Linguistics*, 9:1425–1441.

Gabriel Peyré and Marco Cuturi. 2019. Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning*, 11(5–6):355–607.

Auguste Poiroux, Gail Weiss, Viktor Kunčak, and Antoine Bosselut. 2025. Reliable evaluation and benchmarks for statement autoformalization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17947–17969, Suzhou, China. Association for Computational Linguistics.

Archiki Prasad, Swarnadeep Saha, Xiang Zhou, and Mohit Bansal. 2023. ReCEval: Evaluating reasoning chains via correctness and informativeness. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10066–10086, Singapore. Association for Computational Linguistics.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. PRover: Proof generation for interpretable reasoning over rules. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 122–136, Online. Association for Computational Linguistics.

Swarnadeep Saha, Prateek Yadav, and Mohit Bansal. 2021. multiPRover: Generating multiple proofs for improved interpretability in rule reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3662–3677, Online. Association for Computational Linguistics.

Michael Schlichtkrull, Yulong Chen, Chenxi Whitehouse, Zhenyun Deng, Mubashara Akhtar, Rami Aly, Zhijiang Guo, Christos Christodoulopoulos, Oana Cocarascu, Arpit Mittal, James Thorne, and Andreas

Vlachos. 2024. The automated verification of textual claims (AVeriTeC) shared task. In *Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER)*, pages 1–26, Miami, Florida, USA. Association for Computational Linguistics.

Dominic Schuhmacher, Ba-Tuong Vo, and Ba-Ngu Vo. 2008. A consistent metric for performance evaluation of multi-object filters. *IEEE Transactions on Signal Processing*, 56(8):3447–3457.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018. The fact extraction and VERification (FEVER) shared task. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 1–9, Brussels, Belgium. Association for Computational Linguistics.

Author Index

Agadi, Akshaj, 240
Akiyoshi, Ryota, 152
Amamoto, Takayuki, 152
AN, LI, 167
Ando, Risako, 152

Beliaeva, Aleksandra, 126
Burr, Michael, 136, 167

Dessi, Roberto, 13
Dias-Alexiou, Carolina, 291

Harish, Srinivas, 86
Hernandez, Carlos Toxtli, 136, 167
Higashino, Shinichi, 1

Jansari, Vinita Gangaram, 136, 167

Karaosmanoğlu, Toprak Seda, 312
Karbasi, Kaveh, 268
Karbasi, Kia, 268
Khrulev, Ruslan, 30
Kim, Andrew, 240
Kim, Byungjun, 112
Kochmar, Ekaterina, 201
Kosaka, Norio, 1
Koto, Fajri, 69, 201
Kumar, Dhruv, 216

LaHue, Austin, 136, 167
Lee, Jonghyun, 112

Mandal, Murari, 216
Marina, Maria, 194
Marrese-Taylor, Edison, 291
Marripalapu, Neel, 240
Matsuo, Yutaka, 291
Matsuoka, Hikaru, 101
Maurya, Kaushal Kumar, 201
Mazdarani, Fateme, 42
Mineshima, Koji, 152

Nag, Soumyadeep, 305
Nye, Benjamin E, 136, 167

Osada, Genki, 1

Panchenko, Alexander, 194
Patil, Parth Shrikant, 216
Pottie, Gregory, 268
Proctor, Harris, 167
Puente, Luis David Garcia, 136, 167

Rodolà, Emanuele, 13

Sarkar, Suparnojit, 305
Shah, Sajel, 13
Sharma, Avi, 177
Sharma, Jai Kumar, 254
Sharma, Shrenil Shaun, 177
Sheikh, Md. Faisal, 57
Sinha, Yash, 216
Solano, A., 291
Srivatsa, Kv Aditya, 201

Tonga, Junior Cedric, 201
Toxtli, Carlos, 42

Yamada, Rinta, 152
Yamaguchi, Shuji, 1
Yamasaki, Sakiko, 152
Yang, Edison, 240
Yari, Amir Hossein, 69
Yoo, J. Albert, 285

Zhou, Luca, 13