When Does Structured Knowledge Help Neural Theorem Proving?

Sareh Nabi

AI Researcher · Stanford University

Joint work with

CodeX, the Stanford Center for Legal Informatics



Roland Vogl

Executive Director, CodeX

Lecturer in Law, Stanford Law School

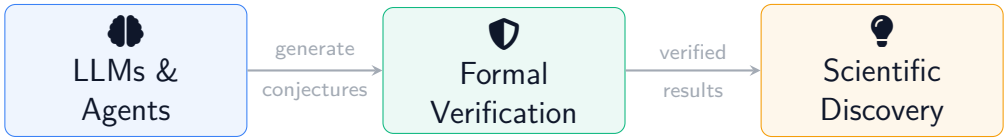


Marzieh Nabi

AI Research and Product Development,

Stanford CodeX and CS

LLMs & Formal Reasoning for Scientific Discovery



Automated Reasoning

LLM agents write proofs, discover patterns, scale exploration

Verifiability

Lean 4, Coq, Isabelle guarantee correctness no hallucination risk

Scaled Discovery

Mathematics, physics, program verification, drug discovery

Our focus: **theorem proving in Lean 4**

Can structured knowledge help LLMs write verified proofs?

The Missing Layer in Formal Theorem Proving

What Mathlib (formal library) has:

- ✔ 285K+ verified theorems
- ✔ Syntactic dependencies
- ✔ Type-checked correctness

What it's missing:

- ✘ Cross-domain analogies
- ✘ Generalization relationships
- ✘ Strategic proof guidance

Current state-of-the-art (2026):

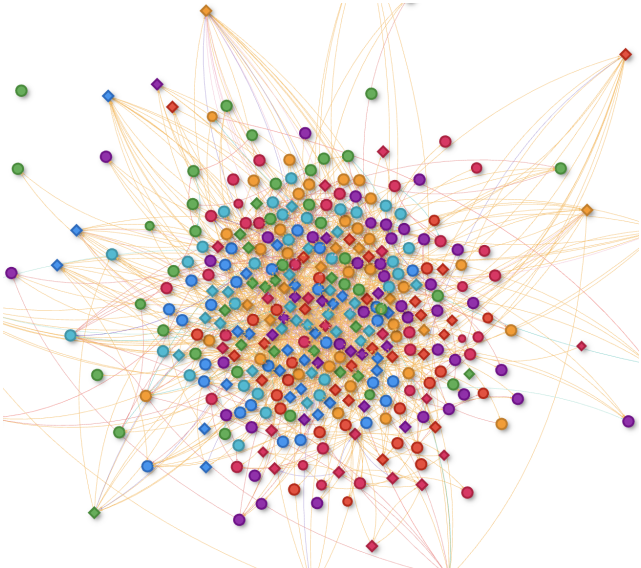
- Leanstral 1.5 (Mistral, 119B MoE): saturates miniF2F, 587/672 PutnamBench
- Seed-Prover 1.5 (ByteDance): 88% PutnamBench, 11/12 Putnam 2025 problems
- Goedel-Prover-V2 (open 8B/32B): 84.6% / 88.1% miniF2F-test

Current state of the art relies on **large-scale fine-tuning and RL** on formal proofs. None systematically evaluate external structured knowledge at inference time.

Research Question

When does structured knowledge at inference time help LLM theorem provers, for which models, and does the answer vary by problem?

MathKG: Semantic Knowledge Graph



364 nodes from Mathlib
229 theorems, 135 definitions

7 domains (49-55 nodes each)
Number Theory, Combinatorics, Algebra, Analysis, Geometry, Linear Algebra, Abstract Algebra

9,434 edges in 8 types
from 132K LLM calls, anchored to verified Mathlib declarations

Interactive graph
sarehnabi.github.io/mathagent

MathKG Schema: Nodes and Edges

Node: name, domain, statement, keywords, **Mathlib declaration name** + import, verification status

Edge: relation type r (one of 8 below), confidence $c \in [0, 1]$, natural-language reason ρ

Edge type ($A \to B$)	**Semantics**	**Restriction**	**Count**
cross-domain-bridge	formal link across domains	different domains	3,481
prerequisite	A needed to state B	A: definition	2,293
applied-in	A's result used in B's proof	both theorems	1,503
analogous-to	same structure, other context	both theorems	578
instance-of	concrete A exemplifies B	both definitions	80
generalizes	A strictly more general than B	both theorems	48
specializes	A is a special case of B	both theorems	30
equivalent-to	$A \leftrightarrow B$, same level	both theorems	13
auto-inverse (symmetric types, gen./spec. pair): 1,408		**Total**	**9,434**

Relation Extraction and Mathlib Anchoring

Relation extraction pipeline (8,026 typed + 1,408 auto-inverse = 9,434 edges)

(1) Evaluate all $364 \times 363 = 132{,}132$ ordered pairs (A, B), since relations are asymmetric

(2) Prompt Sonnet 4.6 with both nodes' statements, keywords, domains, Mathlib names and imports, plus per-type definitions, restrictions, few-shot examples

(3) Output $\{$relationship r (or none), confidence c, reasoning $\rho\}$; discard $c < 0.75$

(4) Post-process: drop type-violating edges and contradictory generalizes/specializes cycles, add inverses

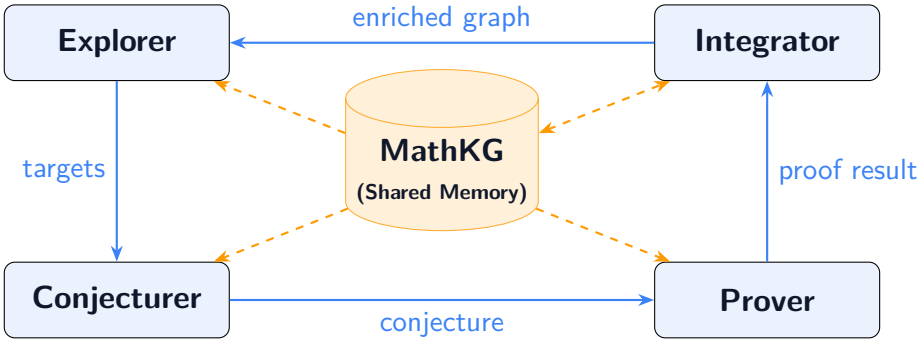
Mathlib anchoring

Every node's Mathlib declaration name is verified against the 404,440-entry Mathlib documentation index, linking it to a Lean 4 type signature; Sperner's theorem links to:

```
theorem IsAntichain.sperner {α :  Type u_2} [Fintype α]
{𝒜 :  Finset (Finset α)}
(h𝒜 :  IsAntichain (fun (x1 x2 :  Finset α) => x1 ⊆ x2) ↑𝒜) :
𝒜.card ≤ (Fintype.card α).choose (Fintype.card α / 2)
```

The linked signature exposes hidden structure; Mathlib retrieval (Layer 2) supplies it to the Prover, which then gets exact names and argument types.

MathAgent: Four LLM Agents and an Orchestrator



Two operation modes (coordinated by the Orchestrator):

- **Open-ended exploration** (implemented, not evaluated): Explorer → Conjecturer → Prover → Integrator
- **Targeted problem solving** (our focus): Explorer → Prover

Targeted Problem-Solving Mode (Algorithm 1)

Require: MathKG $\mathcal{G}$, problem P, max attempts K (default 32)

1: $\text{context}_{\text{KG}} \leftarrow \text{Explorer.search_kg}(P, \mathcal{G})$ ▷ Layer 1: KG semantic search
2: $\text{context}_{\text{Mathlib}} \leftarrow \text{Explorer.search_mathlib}(P)$ ▷ Layer 2: Mathlib declaration search
3: $\text{proof} \leftarrow \text{Prover.generate}(P, \text{context}_{\text{KG}}, \text{context}_{\text{Mathlib}})$ ▷ Attempt 1
4: **for** $k = 2, \ldots, K$ **do** ▷ Error-guided refinement
5: **if** proof.verified **then return** proof
6: $\text{proof} \leftarrow \text{Prover.refine}(\text{proof}, \text{proof.error_log})$ ▷ Previous code + its Lean errors
7: **end for**
8: **return** proof if proof.verified else failed

The four augmentation modes differ only in which of lines 1-2 run:

`prover_only`: neither · `with_kg`: line 1 · `with_mathlib`: line 2 · `full_system`: both

Explorer Agent: Two-Layer Context Retrieval

Layer 1: MathKG semantic search

(1) Pre-filter MathKG to the problem's domain plus its cross-domain-bridge neighbors
(2) Select s=5 seeds by LLM relevance scoring
(3) Walk edges 1 hop from each seed, preserving the edge reasoning behind each connection
(4) Attach to each retrieved node its edges to the seeds (type, confidence, reasoning)

Layer 2: Mathlib declaration search

(1) Extract search terms: LLM-suggested namespace prefixes + regex-extracted identifiers
(2) Prefix-match against the 404,440-entry Mathlib documentation index
(3) Rank by source priority, match precision, name specificity, kind, and module proximity
(4) Enrich the top-ranked declarations with Lean 4 type signatures and docstrings

Full system (1+2): KG block first, then the Mathlib block; declarations in a seed's Mathlib module get a 1.5× boost, and those already in the KG block are dropped.

Evaluation: 5 Models $\times$ 4 Augmentation Modes

Benchmarks (1,520 problems):

MathOlympiadBench	360
miniF2F	488
PutnamBench	672
Total	**1,520**

Protocol: Pass@32, sequential error-guided refinement

4 augmentation modes:

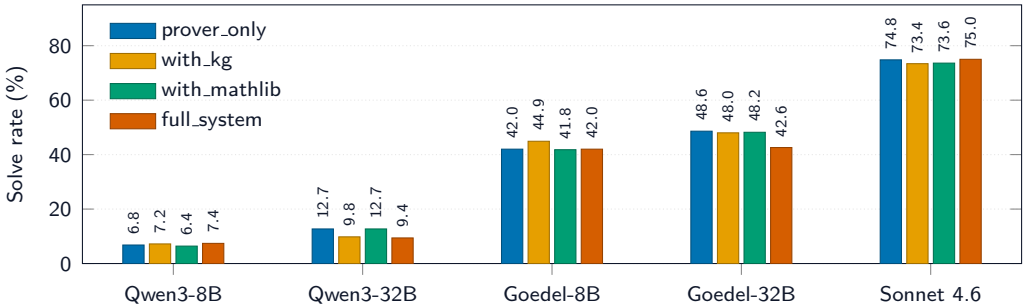
- `prover_only` (baseline)
- `with_kg` (Layer 1)
- `with_mathlib` (Layer 2)
- `full_system` (L1+L2)

Models (matched 2×2 + proprietary): Qwen3-8B/32B (general) vs. Goedel-Prover-V2-8B/32B (Lean-specialized, fine-tuned from the same Qwen3 bases), plus Claude Sonnet 4.6. Cross-model comparison on miniF2F; proprietary model also on PutnamBench and MathOlympiadBench.

Results

Three findings from a controlled ablation
across five models and four augmentation modes

Finding 1: Specialization Dominates Augmentation



+33-38 points

from Lean fine-tuning, in every augmentation mode

8B beats 32B

specialized Goedel-8B leads Qwen3-32B by 29-35 points

$\leq$ 3 points

the most any single augmentation mode adds

miniF2F. External knowledge does not substitute for competence in the weights.

Finding 2: Augmentation Is Capability-Conditioned

Δ problems solved (miniF2F)	Qwen-8B	Qwen-32B	Goedel-8B	Goedel-32B	Sonnet
KG helps? (kg − PO)	+2	−14	**+14**	−3†	−7
Mathlib helps? (mathlib − PO)	−2	0	−1	−2	−6
Synergy? (full − max(kg, mathlib))	+1	−16	−14	−27†	+7

Scale sets the sign.
KG context helps at 8B (+2, +14) but hurts at 32B (−14, −3) and for the proprietary model (−7).

Specialization sets the magnitude.
The Lean-tuned model gains more than its general base at both scales.

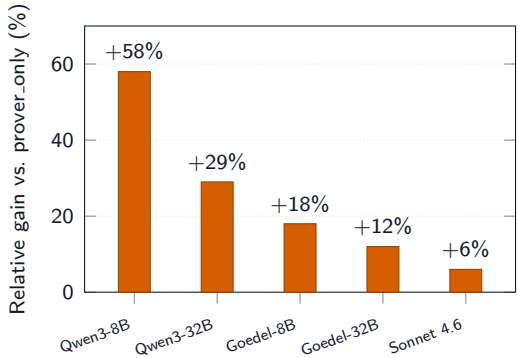
Lexical Mathlib retrieval never helps on aggregate. No fixed mode helps every model.

†Goedel-32B's long generations overflow the 40,960-token window on 10 / 28 problems in these modes.

Finding 3: The Complementarity Effect

miniF2F	PO	Oracle	Gain
Qwen3-8B	33	52	+19 (+58%)
Qwen3-32B	62	80	+18 (+29%)
Goedel-8B	205	241	+36 (+18%)
Goedel-32B	237	266	+29 (+12%)
Sonnet 4.6	365	388	+23 (+6%)

Oracle = union of the four modes,
i.e., the best mode chosen per problem.

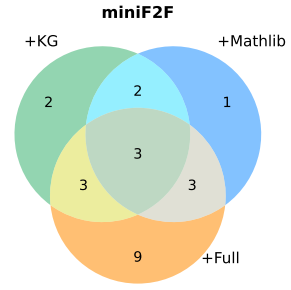


Per-problem selection beats the unaugmented prover for **every** model (+4-7 points), even the strongest, where no fixed mode improves the aggregate. The gain shrinks as capability grows.

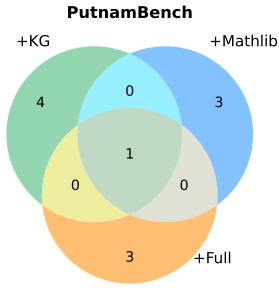
Complementarity Strengthens on Harder Problems

Claude Sonnet 4.6, pass@32, three benchmarks. Oracle = union of the four modes.
Gain = (oracle − prover_only)/prover_only.

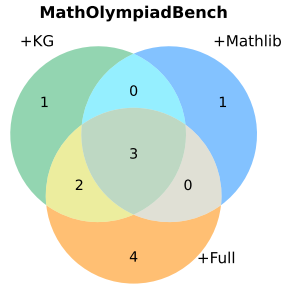
Benchmark	prover_only	with_kg	with_mathlib	full_system	Oracle	Gain
miniF2F (488)	365 (74.8%)	358 (73.4%)	359 (73.6%)	**366 (75.0%)**	388	+6%
MathOlympiadBench (360)	**39 (10.8%)**	33 (9.2%)	36 (10.0%)	38 (10.6%)	50	+28%
PutnamBench (672)	**34 (5.1%)**	30 (4.5%)	31 (4.6%)	25 (3.7%)	45	+**32**%



Baseline: 365 | Oracle: 388 | Lost: 7



Baseline: 34 | Oracle: 45 | Lost: 3



Baseline: 39 | Oracle: 50 | Lost: 3

Summary

1. Specialization dominates augmentation: Lean fine-tuning adds 33-38 points in every mode; no single mode adds more than 3.

2. Augmentation is capability-conditioned: KG context helps small models but hurts large ones, with the specialized model gaining more at every scale.

3. The modes are complementary: per-problem selection solves 6-58% more than the unaugmented prover, and the margin widens on harder problems.

Structured knowledge helps only when matched to the model and the problem
$\Rightarrow$ adaptive proving: choose augmentation per model and per problem.

Thank You

Questions? Feedback?

Appendix

A MathKG Node

Cayley-Hamilton Theorem (Linear Algebra), as stored in MathKG:

```
id: "thm_la_002"
name: "Cayley-Hamilton Theorem"
domain: "Linear Algebra"
type: "theorem"
statement: "Every square matrix satisfies its own characteristic
polynomial."
keywords: ["Cayley-Hamilton", "characteristic polynomial"]
lean4_mathlib_name: "Matrix.aeval_self_charpoly"
lean4_mathlib_import: "Mathlib.LinearAlgebra.Matrix.Charpoly.Basic"
lean4_status: "verified"
```

The Relation Extraction Prompt

Prompt skeleton

Role: classify A → B; "be conservative."

NODE A / B: name, domain, type, statement, keywords, Mathlib name + import.

Ontology: 8 types, each with definition, example, restriction.

"none": "both about X," vague analogies, or confidence < 0.75. "A sparse, high-quality graph beats a dense noisy one."

7 few-shot examples, incl. a "none" case.

Output: JSON {`relationship`, `confidence`, `reasoning`}.

Two real edges from MathKG

Euler's Theorem → Fermat's Little Theorem
`generalizes`, confidence 0.99
"When $n = p$ is prime, $\varphi(p) = p - 1$, so Euler reduces exactly to $a^{p-1} \equiv 1 \pmod{p}$. B is recovered from A by fixing n prime."

First Isomorphism Thm (groups) → (rings)
`analogous-to`, confidence 0.93
"Same schema: the quotient by the kernel is isomorphic to the image. The Mathlib names (`quotientKerEquivRange` vs. `quotientKerEquivOfSurjective`) confirm one schema in two algebraic categories."

Edge Quality, Restrictions and Cleanup

Sonnet 4.6 violates the prompt's stated restrictions in 11.4% of edges (1,041 of 9,141); a post-pass enforces them:

Edge type	Before	Dropped	After	Violation
applied-in	2,277	750	1,527	B is a definition
analogous-to	701	118	583	one endpoint is a definition
cross-domain-bridge	3,631	139	3,492	same domain
generalizes	69	19	50	B is a definition
specializes	37	6	31	B is a definition
equivalent-to	17	4	13	one endpoint is a definition
prerequisite	2,329	5	2,324	A is a theorem
instance-of	80	0	80	none
Total	**9,141**	**1,041**	**8,100**	

Then 74 duplicate-ID edges are dropped $\Rightarrow$ 8,026 raw + 1,408 auto-inverse = 9,434.

Limitation: no human audit yet; quality rests on the 0.75 threshold, restrictions, and cleanup.

What the Prover Actually Receives

Layer 1: MathKG context block

- Domain pre-filter: for a number theory problem, 364 nodes $\rightarrow$ 172 candidates
- The evaluated model scores candidates by relevance; top s=5 become seeds
- 1-hop walk from each seed, keeping the edge reasoning text
- Output order: seeds first, then neighbors sorted by edge confidence, each with statement, relevance rationale, and connecting edges (type, confidence, reasoning)

Layer 2: Mathlib declarations

- Search terms: 10-15 LLM-suggested namespace prefixes (essential / helpful / speculative) plus identifiers regex-extracted from the statement
- Prefix match over the 404,440-entry index (`Nat.Prime` alone matches about 110)
- Five-signal ranking, capped at 8 / 5 / 2 per term; each declaration delivered as:

```
theorem ZMod.pow_card_sub_one_eq_one
{p : ℕ} [Fact (Nat.Prime p)]
{a : ZMod p} (ha : a ≠ 0) :
a ^ (p - 1) = 1
-- Fermat's Little Theorem: nonzero
-- a : ZMod p has a ^ (p - 1) = 1.
```

Full system: Layer 2 declarations sharing a Mathlib module with a Layer 1 seed get a 1.5$\times$ boost; anything already surfaced by Layer 1 is removed, so Layer 2 is strictly complementary.

The Context-Overflow Cost (Goedel-32B)

Goedel-32B writes the longest proofs of the five models. Injected context pushed some attempts past its 40,960-token window; those count as failures.

Goedel-32B, miniF2F	Overflowed	Solved (reported)	Excluding overflow
`prover_only`	0	237 (48.6%)	237/488 (48.6%)
`with_kg`	10	234 (48.0%)	234/478 (49.0%)
`with_mathlib`	0	235 (48.2%)	235/488 (48.2%)
`full_system`	28	208 (42.6%)	208/460 (45.2%)

- Excluding overflow, `with_kg` is neutral vs. `prover_only` (49.0% vs. 48.6%; on the same 478 problems the gap is $+1$): part of the -3 and -27 in Finding 2 is a systems cost, not a reasoning effect.
- Overflow occurred only under augmentation and only for the most verbose model: context competes with the proof for tokens.

Limitations

- **Graph scale.** 364 curated nodes against 285,000+ Mathlib theorems: the results show what a small, curated semantic layer does, not what full coverage would.
- **Edge provenance.** A single LLM (Sonnet 4.6) with a 0.75 threshold and programmatic cleanup. No human audit yet.
- **Fixed KG retrieval settings.** Layer 1 (MathKG) uses $s{=}5$ seeds and one hop with an uncapped context block; pruning the block is untested.
- **Lexical retrieval only.** Mathlib retrieval prefix-matches declaration names; the negative result does not speak to learned premise selection.
- **Untested context selector models.** All context selection (Layer 1 seed scoring, Layer 2 search terms) is done by the evaluated model itself; a stronger selector model is untested and could raise the augmented modes and the oracle gain.
- **Open-ended mode.** Implemented but not evaluated; all results are for targeted proving.