SPECTER LABS PREPRINT

Proto-Cognitive Signatures in Distributed MCTS Theorem Proving

A TAME-style perturbation assay for same-goal, variable-means proof search

Ludwig Pouey

Specter Labs

April 2026

ABSTRACT

We ask a simple question about proof search: if a theorem prover finds a proof and we remove one tactic family from that successful path, can it still reach the same theorem? We test this in a distributed Monte Carlo tree search prover whose local controllers share a search frontier. The experiment is a TAME-style perturbation assay: we use goal preservation under changed means as an operational test, not as a claim about consciousness or biological mind [1, 2]. In the current completed-run lake, 367 of 1,064 baseline-solved intervention rows still solve after the tactic block. In the stricter null-stable slice, 96 solved rows produce a different proof-family hash and 187 show non-zero search-graph distance. The prover therefore sometimes reaches the same theorem by a different route. The failures are just as informative: blocking `linarith` on `hf_deepseek_prover_v1_train_11756` collapses every matched ReProver seed in a fixed 16-seed panel and also fails under DeepSeek, showing a real arithmetic bottleneck rather than a generic tactic preference. Other tactic blocks are often survivable, and some even help by pruning bad search habits. The result is a concrete map of what this prover can route around, what it depends on, and when damage becomes a better search bias.

Keywords. distributed theorem proving, MCTS, collective intelligence, proto-cognition, TAME, proof graphs

1. Introduction

Formal theorem proving is a useful setting for perturbation science because the goal state is explicit, the search trajectory is inspectable, and intervention points are controllable. Unlike many open-ended agent benchmarks, proof search lets us ask a precise question: when a successful trajectory is damaged, does the system fail outright, reproduce the same proof family, or preserve competence by reaching the same theorem through different means?

This paper studies that question in a distributed Monte Carlo tree search system in which multiple local controllers coordinate over a shared frontier. The distributed layer is not introduced as a generic performance booster. Instead, it is treated as a collective-intelligence perturbation mechanism: by changing which local decisions can be taken, when they can be taken, and how information is routed across the frontier, we can probe how flexible the higher-level search process really is [3, 4].

The conceptual lens comes from Michael Levin's TAME program, which argues that cognitive and teleological vocabulary should be evaluated by empirical leverage rather than by metaphysical comfort. On that view, the relevant signature is not surface resemblance to humans, but flexible navigation in a problem space: the ability to reach a fixed goal by variable means under changing conditions [1, 2, 5].

The claim is operational, not metaphysical. We do not argue that theorem provers are conscious, sentient, or biologically equivalent. We ask whether intervention-response signatures in this system support a proto-cognitive interpretation in the TAME sense: some distributed proof-search processes maintain or regain goal-directed competence under perturbation by shifting to alternative proof-graph pathways.

This paper makes three concrete contributions. First, it defines a lesion-based framework for theorem proving that distinguishes `replicate`, `reroute`, and `collapse` responses on baseline-solved theorem-runs. Second, it combines proof-graph families, graph edit distance, trajectory divergence, and basin analysis into a single structural assay for rerouting. Third, it uses that assay to test a TAME-style hypothesis: theorem-runs with broader proof basins should be more likely to preserve goal reachability after perturbation.

2. TAME Framing And Research Questions

Levin's framing is useful here because it keeps the paper disciplined. TAME treats agency language as a hypothesis about fruitful interaction protocols: if talking about a system as goal-directed improves prediction, intervention, or synthetic control, then that language has earned provisional scientific use [1, 2, 6]. The theorem-proving system gives us an unusually clean

environment in which to test that stance, because the objective is crisp and the space of interventions is programmable.

For this paper, the key TAME motif is William James' formulation of intelligence as a fixed goal pursued by variable means. In proof search, the theorem is the fixed goal, while tactic choices, subtree expansions, controller allocations, and proof-graph families provide the variable means. A lesion experiment becomes informative when it forces the system to reveal whether the goal was reachable only through a narrow channel or through a broader basin of alternatives. The paper therefore revolves around four operational questions.

Question 1. When an intervention is applied to a solved theorem-run, how often does the system replicate the original proof family, reroute to a distinct family, or collapse?

Question 2. When reroutes occur, are they structurally real rather than superficial label noise? The answer should show up in proof-graph edit distance, novel-goal counts, and trajectory divergence.

Question 3. Does theorem-level basin structure predict lesion resilience? If broader proof basins support rerouting, that is the strongest bridge from the data to a proto-cognitive reading.

Question 4. Does perturbing the distributed scheduler measurably reshape controller behavior even when theorem-level outcomes are buffered? This matters because the distributed layer should count as a genuine intervention surface, not merely as a different implementation detail.

These questions are falsifiable, and they also define the paper's limits. If lesions mostly produce collapse, if reroutes fail to show structural divergence, or if basin structure does not predict resilience, then the proto-cognitive interpretation should be weakened accordingly.

3. System And Perturbation Design

The experimental system is a distributed theorem prover built around Monte Carlo tree search with LLM-backed tactic proposal. Multiple local controllers operate over a shared frontier, so theorem proving unfolds as a coupled search process rather than as a single monolithic rollout. Each node sees partial tree state; the collective achieves proofs no single node could construct alone. This makes the system a useful testbed for collective competence: local units can be individually limited while the aggregate process exhibits higher-level structure.

The perturbation program has two main arms. The first arm performs **path lesions**: once a control run solves a theorem, the system blocks a tactic or tactic family that lies on the solved path and reruns under matched budgets. The second arm performs **scheduler lesions**: block, delay, or reroute schedules alter which controller can act or when an expansion can proceed.

The two arms target different levels of the system. Path lesions probe proof-space substitutability, while scheduler lesions probe the coordination layer itself.

All comparisons in the main lesion analysis should be conditioned on theorem-runs that the baseline solves. This is not a bookkeeping detail; it is part of the design. Theorems that fail in both the control and intervention conditions do not inform the question of lesion-induced rerouting, because they provide no evidence that the system had usable goal reachability to begin with. For stochastic tactic providers, a stricter null-stable subset should also be reported, requiring the matched `control_null` rerun to solve. Rescue cases, where the intervention solves and the baseline does not, are worth analyzing, but they belong in a separate bucket.

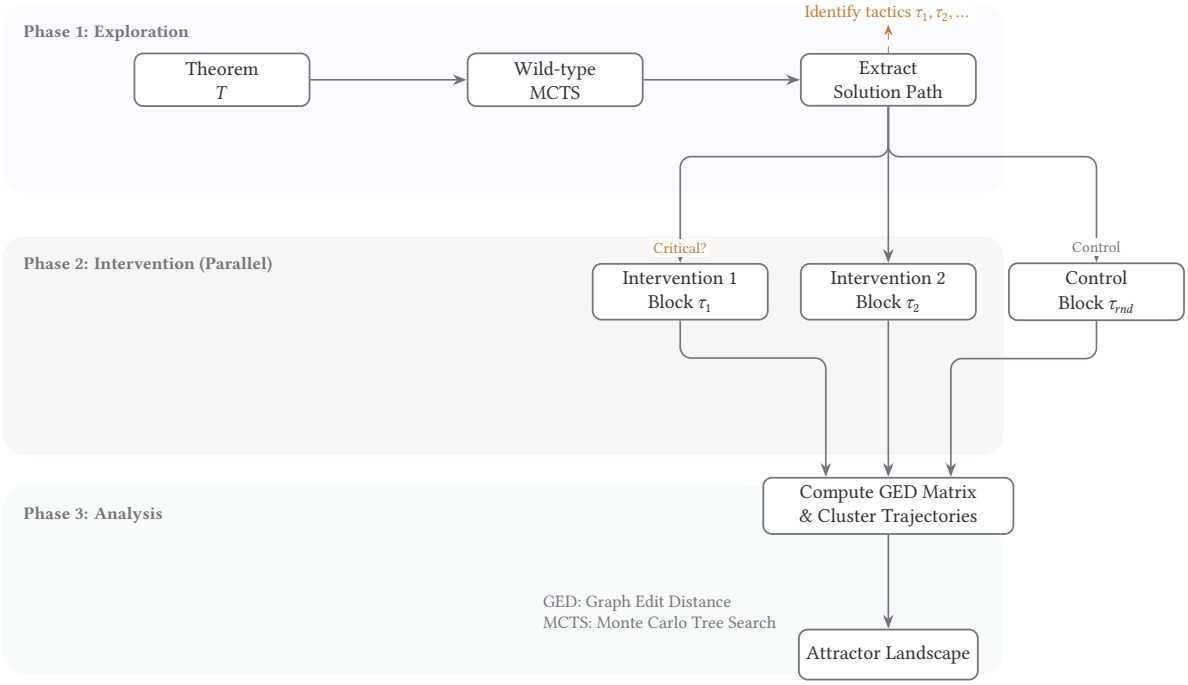

Figure 1: Experimental protocol. Wild-type runs identify solved paths, intervention runs lesion specific tactics or matched controls under fixed budgets, and downstream analysis compares the resulting proof trajectories structurally.

4. Metrics, Inclusion Rules, And Proof Families

The paper should define the denominator before it defines the effect sizes. The main analysis population is the set of intervention rows for which `baseline_solved = true`. A stricter robustness view additionally requires `control_null = solved` for the same theorem-run so that provider stochasticity is not mistaken for lesion collapse. Reporting both views side by side will keep the paper honest, especially if null instability differs materially by provider.

Outcome labels are coarse but useful. `replicate` means the intervention reaches the same proof-family label as the control. `reroute` means the intervention still solves the theorem but lands in a distinct proof-family. `collapse` means the lesion destroys goal reachability within the matched budget. Rescue cases should be reported separately because they are cases where damage helps rather than hurts.

The structural layer of the analysis should make reroutes earn their name. Proof families are defined over proof graphs whose nodes are labeled by goal signatures and tactic families. Graph edit distance provides a coarse family-distance measure, trajectory divergence captures temporal separation from the control path, and recovery metrics quantify whether the system converges back toward the control family or stabilizes elsewhere. These measures allow the paper to distinguish a genuine alternative route from superficial replay noise.

Basin analysis sits one level higher. Repeated control runs on the same theorem define an empirical distribution over proof families, from which we can estimate support counts, entropy, dominant-family mass, and other basin-width summaries. The guiding hypothesis is simple: theorem-runs with wider basins should be more resilient to lesion because they have more accessible structural alternatives.

Element	What this section must lock down
Main denominator	Only theorem-runs with `baseline_solved = true` contribute to the main lesion story.
Stricter subset	Also report the null-stable slice where the matched `control_null` run solves.
Rescue handling	Track `baseline_fail / intervention_solve` as a separate case where damage helps.
Structural metrics	Define proof-family labels, GED, trajectory divergence, recovery, and basin-width measures.

5. Results

5.1. Replicate, Reroute, And Collapse

Each intervention is a break test. We first let the prover solve a theorem. Then we remove one tactic family from the successful path and ask what happens under the same budget. There are three useful answers. The prover may fail. It may solve by roughly the same route. Or it may solve by a different route.

In the current lake, the completed-run cohort contains 338 DeepSeek, heuristic, and ReProver Lean research runs. The main denominator contains 1,064 rows where the unperturbed baseline solved the theorem. After a tactic was blocked, 367 of those rows still solved (34.5%) and 697 failed. The rate differs by provider: DeepSeek recovers in 105 of 290 rows (36.2%), the heuristic provider in 148 of 252 rows (58.7%), and ReProver in 114 of 522 rows (21.8%).

This comparison is only meaningful if the baseline was stable enough to solve again without the lesion. The matched `control_null` run provides that check. It retains 173 of 290 DeepSeek baseline-solved rows (59.7%), all 252 heuristic rows, and 341 of 522 ReProver rows (65.3%). The heuristic provider is therefore much less noisy on this slice than DeepSeek or ReProver. Rows where the baseline failed but the lesioned run solved are a different phenomenon and are handled in Appendix C.

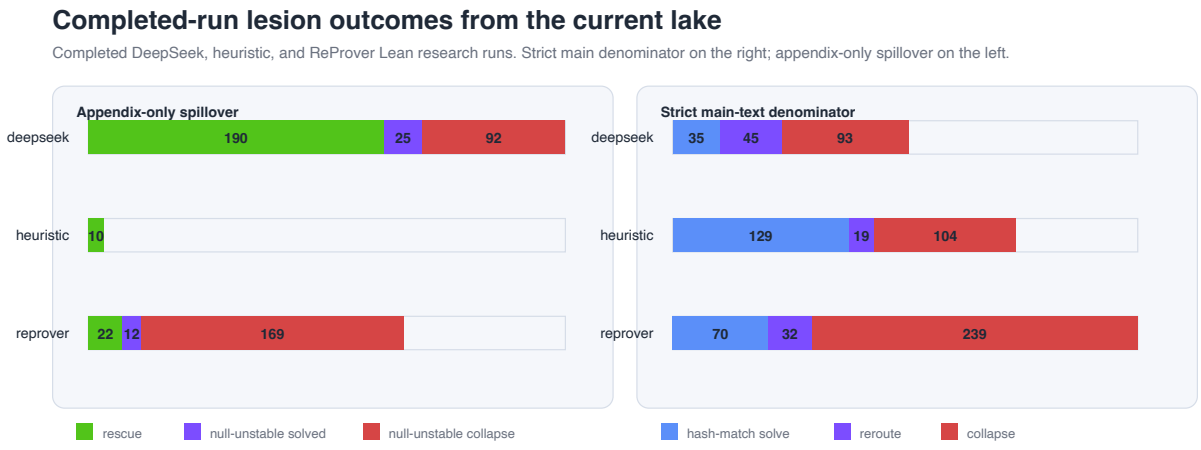

Figure 2: What happens after we block part of a solved proof. Right: baseline-solved rows whose matched `control_null` run also solves. Left: cases outside that strict denominator, including rescues and null-unstable rows.

5.2. When The Prover Survives, The Proof Often Changes

The interesting solved cases are not just lucky repeats. In the strict denominator, 330 rows solve after the tactic block while both the original baseline and the matched null control also solve. Of those 330 rows, 96 (29.1%) produce a different proof-family hash from the control. They reach the same theorem, but not by the same proof.

The weaker structural signal points the same way. Search-graph edit distance is non-zero in 187 of the 330 solved rows (56.7%). The mean normalized distance is 0.362, and the maximum is 0.968. In other words, even when the proof-family hash still matches, the search often took a measurably different path.

Different tactic families fail differently:

Intervention	Total	Recovery	Classification
`block_simp`	59	74.6%	Highly substitutable
`block_apply`	15	60.0%	Recoverable
`block_intros`	108	59.3%	Recoverable entry
`block_intro`	178	47.2%	Mixed entry
`block_cases`	44	45.5%	Mixed structural
`block_exact`	91	34.1%	Often essential
`block_rw`	113	11.5%	Essential
`block_linarith`	52	9.6%	Critical
`block_norm_num`	111	7.2%	Critical numeric

Table 1: What breaks when each tactic family is blocked. Blocking `simp`, `apply`, and `intros` is often survivable. Blocking `rw`, `linarith`, or `norm_num` usually is not.

The table is a map of habits. Blocking `simp`, `apply`, or `intros` often leaves another way through. Blocking `rw`, `linarith`, or `norm_num` usually does not. The entry tactics are in the middle: `intros` recovers in 59.3% of rows, while `intro` recovers in 47.2%. Rewriting and arithmetic closing tactics sit near the hard end.

The reroutes are not free, but they are cheap compared with failure. Reroutes visit more new goals than repeats (0.77 vs. 0.21 on average) and move slightly farther from the original solution path (0.171 vs. 0.153). They also take more iterations than repeats (2.85 vs. 2.26). But collapsed searches average 8.6 iterations, with many more backtracks and goals visited. A successful reroute is therefore a real detour, not a full search blow-up.

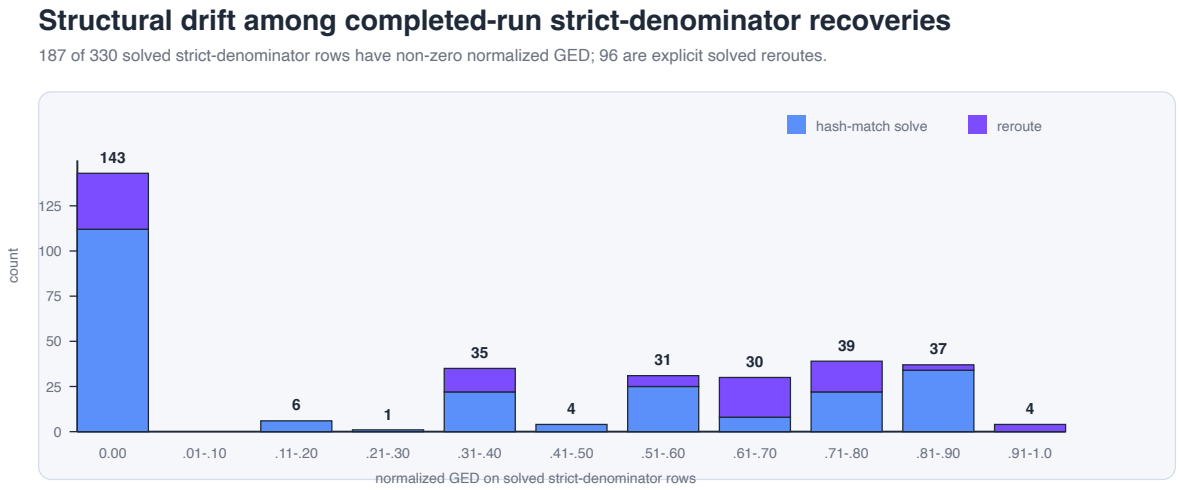

Figure 3: Search-graph distance among solved strict-denominator rows. Of 330 solved rows, 96 are explicit reroutes by proof-family hash and 187 show non-zero graph distance.

5.3. Some Resources Really Are Load-Bearing

Reroutes show that a theorem can sometimes be reached in more than one way. The opposite result is just as informative. Sometimes we remove one resource and every nearby attempt fails. That is a cut: the prover had not learned a flexible route around that resource.

This makes the intervention more than a token ablation. A tactic block is a question asked in the prover's own language: can you still solve the same theorem without this move? If the answer is yes, the move was replaceable. If the answer is no across seeds or providers, the move was carrying real proof work.

The cleanest example is `hf_deepseek_prover_v1_train_11756`, an arithmetic theorem whose solved path uses `linarith`. After a proof-term crash was fixed, ReProver solved the unblocked condition in 16 of 16 seeded runs. With `linarith` blocked, it solved 0 of 16. DeepSeek shows the same shape on the reconciled rows: non-zero wild-type solves, zero `block_linarith` solves. The point is not that arithmetic tactics are generally useful. The point is that this theorem's observed proof route depends on arithmetic compression, and the current search does not find a substitute under the same budget.

The traces make this concrete. In the fixed 16-seed panel, the lake records 128 `linarith` input facts, 96 preprocessed facts, and 32 certificates, plus tactic and rewrite-resource rows. The blocked word is therefore tied to a real compression mechanism in Lean, not just a surface string. Across the lake, the same extraction tracks 226 `simp` used-rule rows, 167 `linarith` fact rows, 123 preprocessed `linarith` fact rows, 37 `linarith` certificate rows, and typeclass-instance traces. Proof-term constant dependencies are stored as DAG edges; the current lake contains 2,478,426 constant nodes and 881,320 constant edges.

Cut	Provider	Outcome	Interpretation
`hf...11756 / linarith`	ReProver	`16/16 → 0/16`	Validated arithmetic chokepoint
`hf...11756 / linarith`	DeepSeek	`6/7 → 0/7`	Cross-provider support
`list_append_nil / cases`	ReProver	`16/16 → 0/16`	Provider-specific chokepoint
`list_append_nil / cases`	heuristic	`16/16 → 16/16`	Substitutable route

Table 2: Cut-sensitivity examples. Some blocked tactics can be replaced. Others remove the only route this prover currently finds.

Provider-specific cuts and provider-stable cuts mean different things. Blocking `cases` on `list_append_nil` collapses ReProver but not the heuristic provider, so that result mostly exposes a ReProver habit. Blocking `linarith` on `hf_deepseek_prover_v1_train_11756` survives the cross-

provider check. That is stronger evidence that the accessible route depends on arithmetic compression itself.

5.4. Basin Width And Lesion Resilience

We expected the broad-basin theorems to be more resilient. If a theorem has many observed proof shapes in repeated control runs, then blocking one tactic should leave more ways around the damage. That prediction is natural, but the current lake does not support it in this simple form.

The lake contains 3,145 basin rows across 2,008 theorem names. Unique-structure counts range from 0 to 21. When we join basin measurements to null-stable lesion outcomes in completed runs, 55 theorem-provider rows remain. In that joined slice, unique-structure counts range from 0 to 5.6, and 11 rows have at least two observed structures. The correlation between unique-structure count and lesion recovery is 0.03. The quartiles are not monotone either: the highest-width quartile recovers at 36.3%, compared with 34.0%, 45.2%, and 39.3% in the lower three quartiles.

The lesson is not that basins do not matter. It is that counting proof shapes is not enough. A theorem can have many variants that all rely on the same arithmetic closer. Another theorem can have only a few observed variants, but those variants may pass through different resources. The cut experiments above explain why this distinction matters: resilience depends on whether the alternatives cross genuinely different proof resources, not only on how many alternatives were observed.

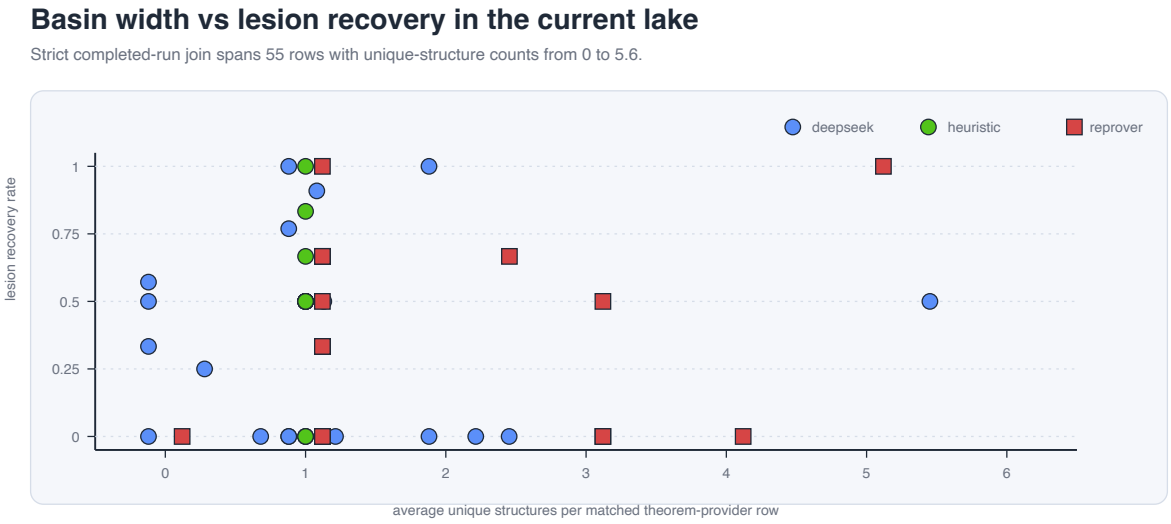

Figure 4: Basin width versus lesion recovery. In this slice, simply counting observed proof shapes does not predict whether a tactic block will be survivable.

5.5. Scheduler Perturbations And Controller-Level Effects

The tactic lesions damage the proof language directly. Scheduler lesions damage a different part of the system: which controller can act, and when. The current completed-run lake contains three March 2026 scheduler-damage runs on ReProver, with block fractions 0.1, 0.3, and 0.5. Each condition has 30 baseline-solved rows and 6 recoveries (20.0%). The raw intervention row totals are 50, 67, and 66.

This is a clean negative result. In this slice, increasing the scheduler block fraction does not reduce theorem-level recovery. The coordination layer is still a real place to intervene, but the sampled theorem outcomes are buffered against this range of scheduler damage. To say more, we need either a larger scheduler matrix or a separate analysis of controller-level traces.

6. Discussion, Limits, And TAME Interpretation

The main observation is simple. When we break a solved proof, the prover often fails, but not always. In 367 of 1,064 baseline-solved rows, it still reaches the theorem. In the stricter null-stable slice, 96 solved rows produce a different proof-family hash, and 187 show non-zero search-graph distance. The prover is not merely replaying the same proof with a missing token. It often finds another path.

Those paths are constrained. Arithmetic and rewriting tactics can behave like hard load-bearing resources. The `linarith` cut on `hf_deepseek_prover_v1_train_11756` collapses every matched ReProver seed in the fixed panel and also fails under DeepSeek. By contrast, other tactics can be removed with little damage, and some lesions even help by pruning bad search habits. The useful picture is therefore not “robust” or “brittle” in general. It is a map of which resources can be replaced, which cannot, and when damage turns into a better search bias.

This is the part that connects to TAME. We are not claiming that a theorem prover is conscious, alive, or mind-like in any broad sense. We are asking a narrower question: does it preserve a fixed goal by changing means when the route is perturbed? In this system, sometimes yes. The proto-cognitive language is useful only to the extent that it helps organize those intervention results: reroute, collapse, rescue, and cut.

The limits are just as important as the positive result. A wider observed basin does not, by itself, predict lesion recovery. The scheduler matrix does not show a theorem-level dose response in the current completed-run slice. Cross-assistant Lean–Rocq comparisons are incomplete. Broader corpora could show that reroutes vanish under tighter controls, that graph distance is mostly noise, or that scheduler damage does not affect theorem outcomes. If so, the interpretation should shrink with the data.

7. Reproducibility

Every number in this build traces to the current Wonton lake at `dossiers/wonton-soup/artifacts/lake/lake.duckdb` (9.9 GB, 857 indexed runs, 338 completed Lean research runs for the main provider cohort, and 28,313 theorem-intervention rows). The figure-generation script queries this lake directly and produces the SVG figures embedded in the paper.

To regenerate figures, run `uv run python paper/build_figures.py --out-dir paper/artifacts` from `dossiers/wonton-soup`. To use another synced lake, set `LAKE_DB_PATH=/local/path/to/lake.duckdb` before that command. To compile the paper, run `nix develop -c typst compile --root ../.. paper/main.typ paper/artifacts/main.pdf --font-path ../../addenda/typst-field-manual/assets/fonts`.

The lake schema and extraction logic are documented in `docs/ops/lake.md` and `analysis/lake/`. Raw run logs are preserved outside the paper source; the lake is the analysis surface used here.

8. Appendix

8.1. Appendix A: Cohort Definitions

Main denominator. A theorem-intervention row belongs to the main denominator if `baseline_solved = true`. This restricts analysis to cases where the unperturbed system solves the theorem, ensuring that collapse or reroute reflects lesion effects rather than baseline failure.

Strict denominator. A subset of the main denominator additionally requires that the matched `control_null` rerun solves. This filters out provider stochasticity: if the system cannot reliably solve the theorem even without lesion, observed failures may reflect noise rather than damage.

Rescue bucket. Rows where `baseline_solved = false` and `intervention_solved = true` are cases where the lesion improves search rather than damaging it. These are excluded from the main analysis and reported separately.

Provider filters. The main analysis includes DeepSeek, heuristic, and reprover providers on the Lean backend in research mode. Coq and other backends are excluded. Runs with partial results or incomplete status are filtered out.

8.2. Appendix B: Lesion Taxonomy

Entry tactics: `intro`, `intros` — appear at proof openings; blocking forces alternative entry points.

Terminal tactics: `rfl`, `trivial`, `decide` — appear at proof leaves; blocking tends to collapse search unless equivalent terminals exist.

Rewriting tactics: `rw`, `simp`, `simp_all`, `simpa` — occupy middle positions; show intermediate recoverability.

Numeric tactics: `norm_num`, `omega`, `linarith`, `ring` — domain-specific closers; blocking collapses search for numeric goals.

Control flow: `constructor`, `exfalso`, `contradiction`, `assumption`, `cases`, `apply`, `exact`, `ext` — structural tactics with varied recoverability.

Scheduler perturbations: `damage-block-f{0.1,0.3,0.5}` — block a fraction of scheduler actions; `damage-delay` — introduce random delays in controller coordination.

8.3. Appendix C: When Damage Helps

Rescue cases—where the baseline fails but the intervention solves—total 222 rows in the completed-run cohort. These rows are useful because they keep the lesion story honest. Blocking a tactic is not always damage. Sometimes it removes a move that was wasting the prover's budget.

Intervention	Rescues	Interpretation
`block_simp`	31	Simplification loops avoided
`block_intros`	28	Plural intro over-commits early
`block_rfl`	17	Forces non-trivial paths
`block_decide`	16	Avoids brittle decision paths
`block_intro`	16	Changes entry tactic commitment
`block_rw`	15	Rewrite chains pruned
`block_apply`	13	Reduces blind application
`block_exact`	11	Forces alternative closers

Table 3: Tactic blocks that most often help after the baseline fails. The largest groups are `block_simp` and `block_intros`.

The largest rescue counts come from blocking `simp` and `intros`. That is a clear warning about search bias. `simp` can pull the prover into simplification work that does not pay off. `intros` can commit too early to a shape of the proof. When those moves are forbidden, the prover sometimes spends its budget elsewhere and finds a proof that the baseline missed.

The rate view adds a second clue. Smaller groups such as `block_have` (34.5%), `block_ext` (12.0%), `block_apply` (8.9%), and `block_exact` (6.3%) rescue less often in raw count but more often within their failed-baseline slices. These are not big enough to carry the main result, but they point to the same mechanism: some tactics are useful in successful proofs and still harmful when the search commits to them at the wrong time.

The theorem-level view makes this concrete. `SubtractionMonoid_x2etoSubNegZeroMonoid_x2eeq_x5f1` has 20 rescues in 48 failed-baseline rows. `add_x5fzsmul` has 19 rescues in 275. That does not mean the original prover was always worse on those theorems. It means that, on those rows, removing a tempting tactic opened a path the baseline did not take.

Bibliography

1. Levin, M.: Technological Approach to Mind Everywhere: An Experimentally Grounded Framework for Understanding Diverse Intelligence, (2022)
2. Levin, M.: Mind Everywhere: A Framework for Conceptualizing Goal-Directedness in Biology and Other Domains, Part Two, (2026)
3. Levin, M.: Collective Intelligence: A Unifying Concept for Integrating Biology Across Scales, (2024)
4. Levin, M.: Chess as a Model of Collective Intelligence: Analyzing a Distributed Form of Chess Play, (2025)
5. Levin, M.: Cognition All the Way Down 2.0: Neuroscience Beyond Neurons in the Diverse Intelligence Era, (2025)
6. Levin, M.: Mind Everywhere: A Framework for Conceptualizing Goal-Directedness in Biology and Other Domains, Part One, (2026)