Why Tactic-Level Rates Cannot Attribute Classical Dependencies in Lean

Vince Gonzalez
ORCID 0009-0005-3640-014X
Draft — 2026-08-08

Abstract

A proof assistant reports which axioms a theorem depends on. It does not report which step introduced them, and a proof invoking several tactics offers no way to apportion the answer. The obvious approach is statistical: score each tactic by how often its proofs carry a classical dependence the theorem did not require, and rank them. This paper reports that the approach cannot work, and gives the reason.

The reason is a mismatch of random variables. Eligibility — a proof reaching Classical.choice where the statement does not — is a property of the theorem. Whether a tactic introduced that dependence is a property of the proof term. A rate conditioned on the first cannot recover the second, and the measurements below show it does not.

Calibration is by known-negatives: tactics that cannot introduce a classical instance, whose rate is therefore background. Across the four Lean libraries containing enough hand-written tactic proofs to measure them, that background is neither small nor constant. In Mathlib it spans 5.8–28.2% across 30 known-negatives; in Lean core 45.0–100.0%, with trivial at the top. The variation is as large within a library as between them, so no normalisation recovers a threshold.

Against those bands, seven candidate escapes occur across four libraries and two controls. Four belong to grind, which is not defective: it proves by refuting the negation, and refutation is classical by construction. One belongs to omega, whose classical dependence is an uncontested measurement — and simp_all ties it at exactly 100.0% in the same cell. None belongs to norm_num, which sits at 5.0% in Mathlib, below the known-negative floor, beneath rfl and exfalso.

That last is the paper's positive result, obtained by a different method. Controlled differential construction — posing the same goal to a tactic and its alternatives and comparing axiom sets — establishes that norm_num closes numeric order goals over $\mathbb{N}$ and $\mathbb{Z}$ through Classical.choice where decide and simp close them without it, by routing through a general PartialOrder lemma on types that carry DecidableEq. Equality goals are unaffected. It is a real, avoidable dependence that every rate in this paper ranks below background.

A third result bounds the ambition. interval_cases scores at the top of its corpus and is innocent: its proofs close with <;> norm_num and carry norm_num's dependence. Attribution to a surface tactic is the wrong unit regardless of the statistic. True attribution is a term-level problem, and is

not solved here.

---

1. The problem, and the mismatch

#print axioms t answers a question about t. A proof of t typically invokes several tactics, each elaborating to a term citing lemmas of its own. When the answer contains Classical.choice, nothing indicates which tactic put it there.

The distinction matters because the remedies differ. A dependence inherent to the statement — the theorem quantifies over the reals, which are constructed with choice — admits no fix. A decision procedure reaching for a classical instance where a constructive one was in scope is fixable once, for everything downstream. A tactic that is classical because refutation is how it works is a property to document.

The mismatch

Write E(t) for t is eligible: its proof reaches Classical.choice and its statement does not. Write D(t, τ) for tactic τ introduced that dependence. A rate estimates

$$P(E(t) \mid \text{choice} \in A(t), \tau \in \text{proof script of } t)$$

while the question is

$$P(D(t, \tau) \mid \tau \in \text{proof script of } t)$$

Eligibility is a property of the theorem; the defect is a property of the proof. A defect that swaps one instance argument inside a helper lemma changes the proof term and leaves the statement untouched — so it cannot move a statistic conditioned on the statement. Worse, τ appearing in a proof script does not establish that its elaborated term caused anything: tactics nest, and §5.1 gives a case where the score belongs to a tactic invoked inside another.

The layers the rate collapses:

```
theorem
├── statement dependency        ← eligibility is decided here
└── proof dependency
└── elaborated term       ← the defect lives here
├── tactic          ← the rate indexes here
│     └── subtactic ← and this is invisible to it
└── lemma / instance machinery
```

Everything below is the empirical consequence.

2. Measurement

2.1 Statement and proof

For a declaration t, A(t) is the set of axioms its proof term reaches and S(t) the union of A(c) over constants c in t's type, both from

Lean.collectAxioms run inside the environment. S(t) ⊆ A(t) holds in all 9,729 measured generated rows.

t is statement-bound in x when x ∈ A(t) and x ∈ S(t); eligible when x ∈ A(t) and x ∉ S(t).

Eligibility identifies candidates whose provenance is separable, not propositions that are constructively provable. A statement whose constants are individually choice-free may still require excluded middle; §5.2 exhibits one. The set is therefore an upper bound on what could be removed, and the word ceiling is used in that specific sense throughout.

2.2 The rate, and its calibration

For tactic τ over proof set P(τ):

rate(τ) = |{ t ∈ P(τ) : t eligible }| / |{ t ∈ P(τ) : choice ∈ A(t) }|

The raw eligible fraction is unusable: a tactic applied mostly to real-analysis goals scores low however it behaves, because its statements are bound regardless. Conditioning on classical proofs removes that.

Calibration is by known-negatives — rfl, exact, apply, intro, rcases, induction, exfalso, trivial and twenty-two others — tactics that cannot introduce a classical instance. Their rate is background by construction. simpa is excluded from this set: it invokes simp. decide is excluded: its status is unresolved (§5.3), so it cannot calibrate anything.

2.3 Controls

Loose. Every tactic named anywhere in the proof script.

Strict. The entire body after := by is a single invocation — no ;, no <;>, no focus dot, no second line — so whatever that call is, it alone closed the goal. 18,230 of Mathlib's proofs qualify.

Both are computed from one join and are comparable by construction. An earlier version of this work defined "single tactic" as *one tactic from a fixed vocabulary*, admitting intro h; decide as lone decide; that is not a control and is not used.

Direct construction. Pose the goal to the tactic and to its alternatives and compare axiom sets. §4 is the argument that this, not any rate, is what establishes causation.

2.4 Corpora and libraries

Four libraries met the prespecified minimum for estimating a background band: at least twenty known-negative proofs. Aesop, ProofWidgets, Plausible, Qq and ImportGraph did not — their declarations recorded as theorems are compiler-generated (322 .sizeOf_spec, 268 .injEq, 164 .mk.inj), and Aesop has four theorem declarations across 137 source files, none proved by tactic. Lean yields sixteen measurable proofs.

| library | proofs | lone proofs |

|---|---:|---:|
Mathlib	80,332	18,230
Init	11,743	4,020
Std	5,116	3,352
Batteries	911	287

A generated corpus is also used: Goedel-Prover output on the Lean Workbook problems, 10,000 proofs, Apache-2.0. Of these, 1 has no parsable theorem header, 270 never entered the environment, and 560 were admitted carrying sorryAx — 831 excluded, 9,169 compiled, 91.7% of the corpus. Lean admits a declaration whose proof failed to elaborate carrying sorryAx, indistinguishable in an axiom report from a proof never finished; those are identified from compiler output. Of the 9,169: 8,496 (92.7%) reach Classical.choice, 7,899 (86.1%) are statement-bound, 597 (6.5%) eligible, 673 (7.3%) choice-free, and none reaches sorryAx, ofReduceBool or ofReduceNat.

That 92.7% → 86.1% → 6.5% cascade is where the statement/proof separation earns its place independently of any attribution: counting classical proofs alone would report 92.7% and mean almost nothing.

Every figure in this paper is produced by one pipeline with one join rule (module plus short name). Earlier drafts contained figures from two pipelines whose name-matching differed, giving two values for the same quantity.

3. The background is wide, and wider within libraries than between them

Known-negative bands, same join, both controls:

library	loose band	median	n tactics	widest
Mathlib	5.8 – 28.2%	16.1%	30	exfalso
Std	0.0 – 60.3%	32.1%	22	assumption
Batteries	23.1 – 81.8%	52.8%	12	refine
Init	45.0 – 100.0%	70.2%	24	trivial

library	strict band	median	n tactics	widest
Mathlib	0.0 – 87.5%	18.3%	7	induction
Init	0.0 – 83.3%	46.9%	3	exact
Std	0.0 – 90.0%	60.0%	3	apply
Batteries	83.3%	83.3%	1	rw

rfl — which cannot introduce a classical instance — scores 16.2% in Mathlib, 28.0% in Std, 48.7% in Batteries and 65.6% in Lean core under the loose control, and 0.0% under the strict control in all three libraries where it reaches the minimum. trivial reaches 100.0% in Init. induction reaches 87.5% in Mathlib under strict — higher than any candidate in that library.

The variation within a library is as large as between them. Mathlib's loose band spans a factor of 4.9 among tactics that provably introduce nothing. This is what closes the obvious repair. One might normalise each tactic by its library's floor; that assumes a single background per library, and there is not

one. positivity at 3.5% is not below background — it is being compared against a background computed on goal populations it never touches. The confound is not between libraries but between the goals each tactic is given, and no library-level constant corrects for it.

4. What the rates do

Candidates against their own library's band. Escapes across all four libraries and both controls:

control	library	tactic	rate	band max	n	is it a defect?
loose	Mathlib	grind	38.8%	28.2%	2,761	no — §4.2
loose	Mathlib	decide	34.8%	28.2%	338	unresolved — §5.3
loose	Std	grind	62.5%	60.3%	89	no
loose	Batteries	grind	83.2%	81.8%	182	no
strict	Init	omega	100.0%	83.3%	66	yes
strict	Init	simp_all	100.0%	83.3%	39	unknown
strict	Batteries	grind	93.2%	83.3%	74	no

Seven escapes. Four are grind, which is architecturally classical and correct as it stands. One is omega — and simp_all ties it at exactly 100.0% in the same library and control, so even there the rate does not isolate it. None is norm_num, established in §4.1 to carry a genuinely avoidable dependence:

tactic, Mathlib loose	rate	position
grind	38.8%	above band
omega	18.9%	inside band
simp	13.4%	inside band
norm_num	5.0%	below band (floor 5.8%)
positivity	3.5%	below band

norm_num ranks beneath rfl, exfalso and every other known-negative.

Neither control rescues the other. Under loose, Mathlib's band is tight enough that grind and decide escape — and grind is not a defect. Under strict, induction at 87.5% widens Mathlib's band until nothing escapes at all. The instrument either fires on the wrong tactic or does not fire.

5. The candidates, individually

5.1 interval_cases, and why the unit is wrong

In the generated corpus interval_cases scores 100% (11 proofs, 8 after removing every proof that touches omega, rate unchanged). It is innocent. Its proofs close with <;> norm_num, and the score belongs to norm_num.

This is not one more rejected candidate. It shows the **unit of attribution is wrong**, independently of the statistic. A surface tactic in a proof script is not the thing that elaborates to a term; a combinator hands the goal to

something else, and the axiom arrives from there. No refinement of a rate indexed on surface tactics can separate the two, because the information is not in that index. Attribution requires following the elaborated term, which is a different measurement than any attempted here.

5.2 tauto, and the meaning of eligibility

tauto scores 100% in the generated corpus (10 proofs) and is innocent. Its flagged proofs are De Morgan's laws for sets:

```
(A ∩ B ∩ C)ᶜ = Aᶜ ∪ Bᶜ ∪ Cᶜ
A ∪ B = A ∩ B → A = B
```

These are classically true and constructively unprovable. Set, ∩ and ᶜ are choice-free constants, so the statement satisfies the eligibility test while the proposition requires excluded middle. Eligibility separates provenance, not constructive provability — which is why it bounds removability from above rather than enumerating it.

5.3 decide — unresolved

measurement	n	rate	position
Mathlib, loose	338	34.8%	above band
Mathlib, strict	77	56.9%	inside band
Init, loose	302	78.8%	inside band
Std, loose	30	26.7%	inside band
generated, loose	113	62.5%	–

Direct construction shows decide closing (2:ℤ) ≤ 4, (2:ℕ) + 2 = 4 and (2:ℕ) ≤ 4 with propext alone — and §4.1 relies on precisely that.

Established: decide produces choice-free proofs for those goals; its aggregate rate varies widely across libraries and controls.
Not established: whether classical dependence in other decide uses comes from the tactic, from the synthesised Decidable instance, or from something beneath that instance.

The plausible explanation is the middle one — the defect one level out, in the instance rather than the tactic. It has not been traced and is not claimed. Until it is, §4.1's use of decide holds for the exhibited goals, where the instances are Int.decLe and Nat.decLe and are verified axiom-free.

6. Controlled differential construction

6.1 norm_num on order goals — avoidable

Lean 4.32.1 with Mathlib:

goal	tactic	axioms
(2:ℤ) + 2 = 4	norm_num	propext
(2:ℤ) ≠ 4	norm_num	propext

(2:ℤ) ≤ 4	norm_num	propext, Classical.choice, Quot.sound
(2:ℤ) < 4	norm_num	propext, Classical.choice, Quot.sound
(2:ℤ) ≤ 4	decide	propext
(2:ℤ) ≤ 4	simp	propext

The same for ≥, >, over ℕ, and with a variable and hypothesis. Every order relation is affected; equality and disequality are not.

```
(2:ℤ) ≤ 4  by norm_num
→ Mathlib.Meta.NormNum.isNat_le_true → Nat.mono_cast
→ monotone_nat_of_le_succ → Nat.rel_of_forall_rel_succ_of_le
→ LE.le.eq_or_lt → eq_or_lt_of_le → lt_or_eq_of_le
→ Classical.propDecidable → Classical.choice
```

lt_or_eq_of_le : a ≤ b → a < b ∨ a = b is stated for a general PartialOrder, where equality is not decidable, so its appeal to a classical instance is correct. The lemma is not the defect. The norm_num order extension runs only on types carrying DecidableEq and inherits the general lemma regardless.

The constructive route was not obtained on the first attempt. Substituting Int.decEq and closing via lt_of_le_of_ne still produced a classical proof, because lt_of_le_of_ne is itself an order-hierarchy lemma carrying the same dependence. Only leaving the hierarchy succeeds:

```
theorem e (a b : ℤ) (h : a ≤ b) : a < b ∨ a = b := by
by_cases hab : a = b
· exact Or.inr hab
· exact Or.inl (Int.lt_iff_le_and_ne.mpr ⟨h, hab⟩)
```

This reports propext alone, as do Int.decEq and Int.lt_iff_le_and_ne. Had the first attempt been treated as the test, the conclusion would have been that no constructive route exists.

The procedure that found it:

```
candidate → construct goals differing only in the relation
→ close each with the candidate and with alternatives
→ compare axiom sets
→ trace the proof term of the differing case
→ exhibit a constructive route, or fail to
```

Each step is falsifiable and none depends on a rate. This is a diagnostic for individual candidate behaviours, not a general attribution framework — it does not scale to a table and does not resolve nesting.

6.2 grind — classical by architecture

grind is the most frequent escape in §4 and is correct as it stands.

goal	grind	term proof
True	classical	choice-free
p → q → p ∧ q	classical	choice-free

grind proves True using the axiom of choice, via

Classical.byContradiction → Classical.propDecidable → Classical.choice. It establishes P by assuming ¬P and deriving False; proof by contradiction is classical by construction.

Worth documenting as a property rather than filing as a fault: a development that cares about constructive content acquires Classical.choice from grind unconditionally, on every goal, including goals provable by trivial.

7. What is established

claim	status	evidence
Statement/proof separation identifies dependencies absent from the statement	established	9,169 generated proofs; 92.7% → 6.5%
Known-negative background varies widely, within and between libraries	established	4 libraries, 2 controls, §3
Tactic-level rates do not reliably identify defective tactics	established in the tested corpora	§4: 7 escapes, 4 of them grind
omega has an avoidable classical dependence	measurement uncontested; characterisation as a defect is the author's	[1]; upstream closed it as a deliberate non-goal without disputing a figure
norm_num has an avoidable dependence on order goals	established	§6.1, direct construction
grind's classical dependence is architectural	established for the exhibited goals	§6.2
decide's dependence is attributable to the tactic itself	unresolved	not traced, §5.3
Surface-tactic indexing is the wrong unit	established	§5.1, interval_cases
Term-level attribution would succeed	not established	not attempted

8. Limitations

The negative result concerns aggregation over tactics, not the statement/proof separation, which §2.4's cascade shows doing real work.

Four libraries is every Lean library meeting the prespecified minimum, but the conclusion is about Lean and Mathlib-style developments, not proof assistants generally.

Attribution here is to a surface tactic. §5.1 shows that unit is wrong; term-level provenance might succeed and is not attempted.

Eligibility bounds removability from above, with a live counterexample in §5.2.

decide is unresolved (§5.3).

Source parsing is syntactic: theorems are matched by module and short name, term-mode proofs excluded, macro- and derive-generated declarations not attributed.

831 of 10,000 generated proofs are excluded; whether the surviving 9,169 differ systematically from them is not established.

9. Reproduction

Corpus: banach1729/goedel-workbook-lean427 (Apache-2.0). Tool: pip install gonzalgo. Environment: Lean 4.32.1 with a contemporaneous Mathlib.

The accompanying archive reproduces every generated-corpus figure with one command and no Lean, corpus download or network access; the derived tactic-per-proof table ships in place of the corpus sources. Each direct construction in §6 is a standalone Lean file printing the axioms of every theorem it declares. Paths are relative to the archive root; it is Apache-2.0.

Proofs are compiled in batches with imports hoisted and -D maxErrors raised on the command line — set_option maxErrors inside a file is ignored, and Lean halts at 100 errors before reaching a report placed after the theorems. Axioms are collected per declaration inside the environment; compile failures are read from compiler output and held out. Library figures were derived by streaming the full declaration dump and recomputing reachability outside Lean, traversed in both directions returning identical sets, reproducing [1].

Controls fixed before measurement: a $\mathbb{N}$ goal closed by omega must show choice in the proof and not the statement; a structural proof must show neither; a goal over $\mathbb{R}$ must show both.

10. References

[1] Gonzalez, V. (2026). *Where Formal Libraries Spend Their Axioms: A Cross-Foundation Measurement, and an Avoidable Classical Dependency in Lean's omega.* Zenodo. https://doi.org/10.5281/zenodo.21769846

[2] Lean 4 and Mathlib, leanprover/lean4, leanprover-community/mathlib4.

[3] Goedel-Prover corpus, banach1729/goedel-workbook-lean427, Hugging Face.

[4] The Kernel Index. https://f-keys.com/gonzalgo/kernel-index/