Which Constant Is Responsible? Dominator Analysis of Classical Dependence in Mathlib

Vince Gonzalez
ORCID 0009-0005-3640-014X
Draft — 2026-08-10

Abstract

61.2% of Mathlib's theorems depend on Classical.choice. Asking which constant is responsible for that dependence is a different question from asking which constants a proof touches, and the two answers differ by a factor of 58 on the first case examined: 116,766 theorems reach the order lemma lt_or_eq_of_le, and 2,018 would stop being classical if it were rebuilt constructively. The rest have another route.

The distinction is exactly dominance. In the dependency graph reversed and rooted at the axiom, severing a constant frees precisely the theorems it dominates, so one dominator tree answers for all 766,564 constants what a separate reachability computation answers for one. Built with the iterative algorithm of Cooper, Harvey and Kennedy, it converges in three passes and reproduces six independently computed severing counts exactly.

Two results follow. First, **60.1% of classically dependent theorems have no responsible constant at all**: their immediate dominator is the axiom itself, so no single declaration gates them and no local repair can reach them. Second, the 39.9% that do have one concentrate in a way that a mathematics library is not usually described in. Outside classical logic itself, the largest sites are a functor category instance (5,271 theorems), hash-map well-formedness in the standard library (4,899), the powerset Boolean algebra (3,430) and string-to-list conversion (1,111). Classical dependence in Mathlib accumulates in its infrastructure rather than in its mathematics.

The dominator construction is standard and is used this way in memory profilers and bundle-size tooling; what appears to be new is its application to a proof-dependency graph rooted at an axiom, and the reading of the resulting subtree sizes as removability weights.

---

1. The question

#print axioms t reports the axioms a theorem's proof reaches. Over a library it reports the same thing many times, and the aggregate has a shape: 324,510 of Mathlib's 530,348 theorems reach Classical.choice.

That number does not say where the dependence enters, and the obvious refinement does not either. Counting how many theorems reach a given constant measures traffic, not responsibility. A constant on the path from a great many theorems to the axiom may be carrying none of them, if each has an independent route.

The question with an actionable answer is counterfactual: **if this constant were rebuilt so that its own proof no longer reached the axiom, how many theorems would stop being classical?**

2. Reach is not responsibility

Severing a constant's outgoing edges and recomputing which theorems still reach Classical.choice:

constant	theorems reaching it	freed by severing	ratio
Classical.propDecidable	317,919	91,858	3.5
Classical.byContradiction	285,622	23,550	12.1
Classical.em	318,404	476	669
lt_or_eq_of_le	116,766	2,018	57.9
eq_or_lt_of_le	—	1,221	—
LE.le.lt_or_eq	—	583	—

lt_or_eq_of_le — a ≤ b → a < b ∨ a = b — is on the path from 36.0% of the library's classical theorems and uniquely responsible for 0.6% of them.

Classical.em is the sharpest case. 318,404 theorems reach it, more than reach propDecidable, and severing it frees 476. The reason is architectural: Classical.propDecidable reaches the axiom directly rather than through em, so removing em leaves everything standing that propDecidable supports. A reachability ranking puts em at the top of the library; a counterfactual puts it below four order lemmas — lt_or_eq_of_le, eq_or_lt_of_le, LE.le.eq_or_lt and LE.le.lt_or_eq — each of which frees more theorems than it does.

3. Dominators

Write the dependency graph with an edge n → d when declaration n names constant d in its type or its proof term, and reverse it. In the reversed graph rooted at Classical.choice, a constant c dominates n when every path from the root to n passes through c — equivalently, when every path from n to the axiom in the original graph passes through c. Severing c therefore frees exactly c's dominator subtree, and the subtree size is the counterfactual of §2 computed for every constant at once. 398,967 declarations lie on some path to the axiom and form the tree.

This is the standard construction. Lengauer and Tarjan gave the classical algorithm; the reading used here — that a dominator subtree is the cost wholly attributable to its root — is how memory profilers attribute retained heap and how bundle-size tools attribute dependency weight. The iterative formulation of Cooper, Harvey and Kennedy is used, which needs no auxiliary structures and converges in three passes on this graph.

Chains are collapsed before reporting. A run of constants each dominating the next with no theorems lost between them is one site, since severing any member frees the same set; counting them separately counts one repair several times. Collapsing at a 0.97 weight ratio takes 331,009 constants with a

non-empty subtree to 327,981 sites. Std.DHashMap.Internal.Raw.WF.out, wfImp_alterₘ and isHashSelf_updateBucket_alter carry 4,892, 4,896 and 4,899 and are one site, not three.

Related measurements

Mendoza-Smith partitions Mathlib by transitive dependence on Classical.choice and stratifies it by distance from the axiom in the dependency graph, reporting that a learned representation separates classical from constructive proofs sharply at distance 2 and not at all beyond distance 9. Distance and dominance are both graph statistics on the same edges and they answer different questions: distance measures how far a proof sits from the axiom, dominance measures whether anything between them is load-bearing. A constant can be at distance 2 and free nothing, which is the case for Classical.em in §2. Neither measure subsumes the other, and the present note makes no use of distance.

Fan and DeDeo's tactic ablation removes non-constructive solution paths and observes which proofs survive. That is an intervention on the prover; severing here is an intervention on the graph, and models a repair to one declaration rather than the withdrawal of a tactic.

Li, Peng, Severini and Shafto map Mathlib's dependency graph at comparable scale and separate compiler-synthesised edges from proof-driven ones. They carry no axiom layer, and that separation is not applied here — §8 notes it as the most obvious refinement available.

The dominator construction itself is standard, and this note claims no part of it. What appears not to have been done before is its application to a proof-dependency graph rooted at an axiom, with subtree size read as a removability weight.

4. Validation

The dominator tree is checked against the severing computation of §2, which shares no code with it: the severing runs recompute reachability from scratch with the candidate's edges removed, while the tree is built once from the graph alone.

constant	dominator subtree	independent severing
Classical.propDecidable	91,858	91,858
Classical.byContradiction	23,550	23,550
lt_or_eq_of_le	2,018	2,018
eq_or_lt_of_le	1,221	1,221
LE.le.lt_or_eq	583	583
Classical.em	476	476

Six of six agree exactly.

The extraction and analysis were written for this note and share no code with the earlier study of the same library, so three previously published figures serve as a second check:

	this note	published
theorems	530,348	532,605
reach Classical.choice	324,510 (61.2%)	324,808 (61.0%)
eligibility ceiling	13.1%	13.1%
dependency edges	29,469,817	30,015,601

No figure moves by more than 3%, consistent with Mathlib version drift.

5. Most classical theorems have no responsible constant

Charging each of the 324,510 classically dependent theorems once, to its immediate dominator, partitions them exactly.

195,008 of them — 60.1% — have Classical.choice itself as immediate dominator. No constant gates them: they reach the axiom by two or more independent routes, and severing any single declaration leaves them classical. Only 39.9% have a proximate constant that could be named as responsible.

This is a ceiling on local repair, and it sits above the eligibility ceiling rather than replacing it. Eligibility bounds how many theorems could be stated without choice; this bounds how many could be reached by fixing one thing.

Charged by area, each theorem counted once:

area	theorems	share
classical logic	239,570	73.8%
category theory	11,879	3.7%
order and root namespace	11,070	3.4%
Std, other	5,976	1.8%
Std, hash maps	5,057	1.6%
sets	4,102	1.3%
lists	2,154	0.7%
filters	1,735	0.5%

6. Where the dependence sits

Ranked by subtree size — how many theorems rest on the site alone. Subtrees nest, so this column is not a partition and does not sum to the library.

theorems	kind	site
91,858	def	Classical.propDecidable
23,550	thm	Classical.byContradiction
5,271	def	CategoryTheory.Functor.category
4,899	thm	Std.DHashMap.Internal.Raw.WF.out
4,711	def	Classical.indefiniteDescription
3,430	def	Set.instCompleteAtomicBooleanAlgebra
2,018	thm	lt_or_eq_of_le
1,484	thm	Set.image_univ
1,290	def	GroupWithZero.toDivisionMonoid

1,221	thm	eq_or_lt_of_le
1,179	thm	LE.le.eq_or_lt
1,112	def	CategoryTheory.Functor.comp
1,111	def	String.toList
1,044	def	AddAction.orbitRel
1,009	def	QuotientAddGroup.leftRel

Two observations.

Outside classical logic, the load-bearing sites are infrastructure. A functor category instance, hash-map well-formedness, the powerset Boolean algebra, string-to-list conversion, the orbit relation of an additive action. None is a theorem anyone would describe as a piece of mathematics. The four largest of them carry 14,711 theorems between them; the three order lemmas in the same table carry 4,418.

They are mostly instances rather than lemmas. Of the fifteen sites above, nine are definitions — typeclass instances and structure projections. This is the layer at which a proof term reaches for machinery the goal did not mention.

A caution on the two largest. 98.0% of choice-dependent theorems reach Classical.propDecidable and 98.1% reach Classical.em, which is closer to architecture than to discovery: Lean derives classical logic from choice, so classical reasoning routes through these by construction. The informative residue is the complement — **6,591 theorems, 2.0%, reach Classical.choice without reaching propDecidable**, and use the axiom for choosing rather than for excluded middle.

7. Where choice is spent

Dominance locates what theorems rest on; it does not locate where the axiom is actually invoked. A site's own route to the axiom runs through its dominator ancestors, not its subtree, so looking inside a site's subtree for the invocation looks in the wrong direction: of the twenty largest sites, fifteen contain no declaration that names a classical primitive at all.

Counting declarations that name one directly in their own type or body, **8,354 name Classical.propDecidable and nothing else**. These are the sites where a substitution test applies, since the classical instance is present as a literal argument and a constructive one may be synthesisable in its place.

A sample of the twenty such declarations with the largest dominator subtrees was tested by re-synthesising Decidable p for each occurrence of Classical.propDecidable p, inside the declaration's own local context. Of 114 occurrences, 21 are closed once the leading binders are introduced and can be synthesised in isolation; the other 93 sit under inner binders and are not testable by this method. Of the 21, synthesis succeeded for 8, and **all 8 synthesised instances are axiom-free**. The propositions involved are arithmetic over Nat and Int — x % 2 = 0, b < c, (2 ^ (n + 1) - (x + 1)) % 2 = 1, ↑n = 0.

That is consistent with the previously reported pattern of a classical Decidable instance being supplied on types where a constructive one exists,

but eight resolved occurrences is a sample and not a rate. No claim is made here about how far it generalises, and none of the eight substitutions was rebuilt and kernel-checked, which is what would be required to show the proof survives.

8. Limitations

The counterfactual is a graph property. Severing a constant models a constructive rebuild that may not exist; the subtree size bounds the payoff of attempting one and does not assert it is attainable. Classical.propDecidable is the standing example — 91,858 theorems rest on it alone and it cannot be rebuilt, because it is the classical assumption.

Dominance is measured over the union of statement and proof edges, matching what collectAxioms follows. A separate treatment keeping the two apart would give a different and probably more useful tree, and is not attempted here.

The library is one version of one library. Nothing here establishes that the concentration in infrastructure is a property of formal libraries rather than of Mathlib at this revision.

9. Reproduction

The declaration graph is extracted from the Lean environment with statement and proof dependencies in separate columns, streamed to a file handle: holding 29 million edges in memory exhausts it, and #eval output through a shell redirect is buffered until elaboration ends, so a long run reports nothing. ConstantInfo.value? must be passed allowOpaque := true or theorem proofs read as empty and every statement-versus-proof figure computed from the dump is wrong rather than imprecise.

The dominator tree takes 42 seconds over the full graph; each severing run takes about 20 seconds; the extraction takes roughly three minutes and produces a 594 MB dump.

Environment: Lean 4.32.1, Mathlib v4.32.1.

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] Gonzalez, V. (2026). *Why Tactic-Level Rates Cannot Attribute Classical Dependencies in Lean.* Zenodo. https://doi.org/10.5281/zenodo.21853489

[3] T. Lengauer and R. E. Tarjan, "A fast algorithm for finding dominators in a flowgraph", ACM TOPLAS 1(1), 121–141, 1979.

[4] K. D. Cooper, T. J. Harvey and K. Kennedy, "A Simple, Fast Dominance Algorithm", Rice University, 2001.

[5] E. Martin, "The dominator tree of a dependency graph", neugierig.org, 2023.

[6] R. Mendoza-Smith, "Geometric Measurements of the Axiom of Choice in Neural

Proof Embeddings", arXiv:2606.28572, 2026.

[7] Z. Fan and S. DeDeo, "Ablation and the Meno: Tools for Empirical Metamathematics", arXiv:2604.22519, 2026.

[8] X. Li, N. Peng, S. Severini and P. Shafto, "The Network Structure of Mathlib", arXiv:2604.24797, 2026.