Machine-Checked Economics: When Computers Verify Proofs

Jon Smirl

2026-03-01

The Problem with Proofs on Paper

Economics papers are full of mathematical proofs. Propositions, lemmas, theorems – they form the logical backbone of theoretical work. But here is an uncomfortable fact: these proofs are checked by humans, and humans make mistakes.

A referee reading a 40-page paper with a dozen proofs has limited time. They check the logic as carefully as they can, but subtle errors slip through. Sign mistakes, unstated assumptions, edge cases that invalidate a step – these are not hypothetical concerns (Ioannidis2005). In economics specifically, there is no systematic audit of theoretical results. The field trusts that peer review catches errors. Often it does. Sometimes it does not.

What if you could make proof errors structurally impossible?

What Is a Proof Assistant?

A **proof assistant** is software that checks mathematical reasoning step by step (deMoura2021). You write your definitions, theorem statements, and proofs in a formal language. The computer then verifies every logical step – not by running examples or simulations, but by checking the actual chain of deduction against the rules of logic.

Think of it this way. A spell-checker catches misspelled words by comparing them to a dictionary. A proof assistant catches logical errors by comparing each inference step to the axioms of mathematics. If the proof compiles, it is correct. Not "probably correct" or "correct assuming the referee did not miss anything." Correct, period.

The proof assistant we use is **Lean 4**, developed by Leonardo de Moura and collaborators at Microsoft Research (deMoura2021). Lean 4 is backed by **Mathlib**, a community-maintained library of over 100,000 formalized mathematical results. When a Lean proof invokes a fact from Mathlib, that fact has itself been machine-verified. The entire chain of reasoning, all the way down to the axioms of set theory, is checked.

What We Verified

The entire CES framework – every theorem described in the articles on this site – is formalized in Lean 4. The numbers:

- **99 marquee theorems** verified: these are the main results across all papers, from *CES emergence* to the *quadruple role of curvature*, from the CES potential framework to hierarchical architecture.

- **0 sorry**: In Lean, `sorry` is a keyword that marks an unproved claim. It tells the compiler "trust me on this one." Our codebase contains zero instances of `sorry`. Every single claim is fully verified. This is rare even in mathematics formalization projects – most have at least a few `sorry` placeholders.
- **3 axioms**: We take exactly three mathematical facts as given, without proving them in Lean. Each is a classical result that the mathematics community accepts as established.

The Three Axioms

An **axiom** in a formalization is a statement accepted without proof. Using axioms is not cheating – it is a deliberate choice to draw a line between "results we prove ourselves" and "results we rely on from the literature." The key is that axioms are *explicit*. Anyone can see exactly what we assume.

Our three axioms are:

1. The Kolmogorov-Nagumo theorem (1930). This classical result (Kolmogorov1930) characterizes all continuous, symmetric, associative mean functions. It states that any such mean must be a "quasi-arithmetic" mean – a generalized average computed through some continuous, strictly monotone transformation. This is the starting point for proving that CES is the unique consistent aggregation function.

2. Aczel's functional equation (1948). Janos Aczel (Aczel1948) proved a uniqueness result for functional equations arising from bisymmetry conditions. Combined with the Kolmogorov-Nagumo theorem and the economic requirement of constant returns to scale, this pins down CES as the only possibility. Together, these two axioms support the *emergent_CES* result: CES is not assumed but derived.

3. Neishtadt's transition duration scaling. This result from geometric singular perturbation theory (Neishtadt1987) describes how long a dynamical system takes to pass through a bifurcation point. We use it to derive crisis duration predictions. The result is well-established in applied mathematics but would require substantial infrastructure to prove from scratch in Lean.

Everything else – superadditivity bounds, correlation robustness, Nash stability, spectral gap results, the CES potential, conservative-dissipative dynamics, hierarchical ceilings, damping cancellation – is proved from definitions using Lean and Mathlib. No hidden assumptions.

Why Formalization Matters for Economics

Formalization addresses a problem that empirical reforms like pre-registration cannot touch: **theoretical degrees of freedom**.

In empirical work, the replication crisis has led to pre-registration, where researchers commit to their analysis plan before seeing the data. This prevents the kind of post-hoc specification searching that produces spurious "discoveries" (Ioannidis2005). But theoretical work has an analogous problem. A modeler makes many choices – functional forms, boundary conditions, regularity assumptions – and these choices affect which theorems can be proved. Without formalization, it is difficult to tell whether a result follows from deep economic logic or from a convenient but unnecessary assumption.

Formalization forces every assumption into the open. When you write a theorem in Lean, you must state every hypothesis explicitly. There is no room for "it is easy to see that" or "by a standard argument." If the argument is standard, you cite the Mathlib lemma. If the hypothesis is needed, you state it. The compiler enforces transparency.

Consider the *superadditivity_quantitative_bound* theorem. The informal statement says that CES with complementary inputs produces a diversity premium proportional to curvature K times input heterogeneity. But what exactly are the hypotheses? Does the result require strict complementarity ($\rho < 0$), or does it hold at the boundary? Does it require all inputs to be positive? Equal weights or general? In a paper proof, these details might be glossed over. In Lean, they are pinned down to the last quantifier.

What Formalization Does Not Do

Formalization verifies that theorems follow logically from their premises. It does not verify that the premises are economically sensible. If you formalize a model with absurd assumptions, you get machine-verified absurd conclusions.

This is why formalization and empirical testing are complements, not substitutes. The CES framework pairs machine-checked theory with pre-registered empirical tests – 38 predictions tested against real data. Formalization closes the theoretical channel of error; empirical testing closes the empirical channel. Together, they provide a level of assurance that neither can achieve alone.

The Bigger Picture

Machine-checked economics is not just a quality-control exercise. It changes how theory can be built.

Because Lean proofs are modular, results can be reliably composed. If you prove that CES emergence implies certain curvature properties, and separately prove that curvature properties imply certain dynamic behaviors, you can chain these results together with confidence that no subtle inconsistency lurks at the boundary. The CES framework has 97 files organized into eight topic areas – Foundations, Curvature Roles, Potential, Dynamics, Hierarchy, Entry-Exit, Macro, and Applications – and every cross-reference between them is compiler-verified.

This is particularly valuable for the *hierarchical architecture*, where results at one level of the economy feed into results at the next level. The *eigenstructure bridge* connecting technology (the CES potential Φ) to welfare (the welfare loss function V) involves calculations that span multiple levels and timescales. Getting this right on paper is treacherous. Getting it right in Lean is mandatory – the proof does not compile otherwise.

Try It Yourself

The full Lean codebase is open source. You can download it, run `lake build`, and watch the compiler verify every theorem. If it compiles with zero errors and zero `sorry`, the proofs are correct. You do not need to trust us. You do not need to trust the referees. You only need to trust the Lean compiler.

Economics has long aspired to mathematical rigor. Machine-checked proofs deliver – not through rhetoric, but through code that either compiles or does not.

References