Econlib: A Formal Library for Economic and Political Theory

Daniel Lyng
July 8, 2026

Abstract

This paper introduces `Econlib`, a Lean 4 library for economic and political theory. `Econlib` provides reusable formal infrastructure for preferences, probability, optimization, game theory, mechanism design, social choice, and general equilibrium, built on a supporting mathematics layer that supplements `Mathlib` . Its contents include machine-checked versions of central results such as Arrow's impossibility theorem, Gibbard–Satterthwaite, Nash and Bayesian Nash equilibrium existence, VCG incentive compatibility, Arrow–Debreu existence, and the first and second welfare theorems. The paper situates `Econlib` within prior and concurrent work on formal economics and political science, describes how the library can support research workflows (including language-model-assisted formalization), presents the library's design principles, and argues that formalization is valuable not only as a correctness guarantee but as a means of making theoretical work more cumulative: Theorem statements expose scope conditions, shared definitions reduce semantic drift, and reusable proof components allow later models to import established machinery rather than reconstruct it. To address the gap between formal verification and substantive faithfulness, `Econlib` is accompanied by worked examples and semantic tests that exercise public results on recognizable models.

Keywords: formal political economy, formal theory, social choice, economic theory, Lean, formalization, mathematical modeling

JEL Classification: C63, C70, D70, D71, D72, D80

1 Introduction

Over the past three decades, the empirical social sciences rebuilt their standards for credibility. In economics, the credibility revolution remade how findings are identified and reported (Angrist and Pischke 2010). The data and code accompanying a paper is expected to be deposited, rerun, and audited prior to publication (Vilhuber 2020) and a claim is expected to survive independent replication before it is believed (Christensen and Miguel 2018). Political science has adopted similar requirements for the availability of data and analysis. Indeed, replication and open access to data are professional norms (King 1995) to the point where data access and replication is part of the American Political Science Association's ethics guide (Lupia and Elman 2014).

However, the deductive core of both fields—economic theory and formal political theory—has been largely untouched by these developments. Whereas the way quantitative analysis is confirmed has entirely transformed over the last few decades, a theoretical result is still certified much as it was a century ago: An author writes a proof in prose, referees read it, a seminar audience probes it, and the profession extends trust. There is no shared artifact a later reader can rerun and no simple check (beyond rederiving the proofs) that the stated assumptions in fact deliver the stated conclusion. This is surprising because, in principle, deductive theory is more amenable to reproducibility than empirical work. While replicating empirical analysis requires access to raw data, the processing steps taken to normalize the data, and the statistical routines (and presentation code) necessary to reproduce the claims and figures in a paper, deductive theory requires only the definitions, assumptions, and logical steps necessary to derive the conclusion. These are naturally expressed in formal language such that they can be verified by machine.

As will be discussed in Section 3, correctness is not the primary advantage of machine verification. Nonetheless, machine verification provides a valuable correctness guarantee by eliminating an entire class of errors arising from unsatisfied assumptions, subtle non-sequiturs, and otherwise incorrect proofs. The repeated (and belated) identification of flawed proofs shows that this benefit is not merely hypothetical. Arrow's impossibility theorem, a marquee result in social choice, is the canonical case: The proof in the first edition of *Social Choice and Individual Values* (Arrow 1951) was flawed to the point where the theorem as stated was false under the given conditions. The gap was identified several years later by Blau (1957), and Arrow repaired the theorem only in the second edition (Arrow 1963). Later proofs of Arrow's theorem published by Geanakoplos (2005) were found to rest on an incomplete case analysis that had survived both refereeing and a published revision (Nipkow 2009).[1] More recently, the matching-with-contracts framework of Hatfield and Milgrom (2005) was shown eight years later to depend on an unstated "irrelevance of rejected contracts" condition without which stable allocations need not exist (Aygün and Sönmez 2013), and Econometrica has published corrigenda both for a counterexample that was itself incorrect (Levy and McLennan 2015) and for a contracting theorem that quantified over equilibria which, under the paper's stated hypotheses, need not exist (Citanna et al. 2023).

[1]Notably, the gaps in this proof were identified by machine verification using Isabelle/HOL.

Beyond this direct motivation, formalization makes theory more cumulative, accessible, and extensible, much as replication packages have for empirical research. Replication packages are useful in day-to-day research not because they catch fraud or reveal failed replications, but because they let scholars entering a related topic inspect and adapt existing data, code, and measurement choices rather than starting from scratch. The absence of an analogous artifact for deductive theory prevents scholars from reusing results with confidence. The conditions a theorem needs to satisfy are scattered across the setup, the proof, and the field's folklore, forcing anyone who wants to apply it elsewhere to re-derive its dependencies. Without a shared base to build on, each paper must re-establish its underlying machinery—fixed-point arguments, envelope formulas, stochastic orders—or, more often, simply handwave over it. Worse, definitions drift and names are overloaded, so the same term can denote subtly different concepts across papers. Perfect Bayesian equilibrium (PBE) is a case in point: Because no canonical formal definition exists for general extensive-form games, papers differ in the off-path belief-consistency requirement they impose, so results stated under the same name need not compose.

The effect is that scholars building on established results may inadvertently rely on conditions that, while holding in the applications the original author had in mind, do not hold for their envisioned models. This is an easy mistake to make because models begin with familiar objects —a preference relation, a voting rule, a stochastic process, an extensive-form game, an exchange economy—and end with recognizable conclusions about welfare, equilibrium, or collective choice. Between those endpoints lies a chain of mathematical obligations easy to lose track of: Choice sets must be nonempty and compact, integrals must be well-defined, posterior beliefs must be normalized, strategies must respect information sets, and fixed-point correspondences must satisfy the hypotheses they invoke. Most of the time, eliding over these conditions is harmless and aids exposition, but some of the time, the conditions meaningfully restrict the generality of a claim or undermine it completely. Consider a standard existence argument, which typically runs a payoff through a fixed-point theorem that requires continuity. In an all-pay auction—the model behind lobbying, campaign spending, and other winner-take-all contests—each bidder's payoff jumps where bids tie, so the standard existence argument does not apply.[2] In such a model, "an equilibrium exists by a standard argument" is not a harmless abbreviation but a false claim, and the unstated continuity condition is what concealed it.

Ordinary mathematical prose is a weak instrument for tracking obligations of this kind. This is not because theorists are careless but because the assumptions that matter are often local, technical, and scattered across a long argument while the reader's attention is on the substance. While the usual safeguard of peer review is indispensable, it does not make theorem dependencies, side conditions, or edge cases easily and independently inspectable or reproducible. Empirical work met the analogous problem of results that could not be reproduced by building the shared infrastructure of data repositories, trusted statistical software, and replication packages. Deductive theory has not built the equivalent.

[2]The failure of a sufficient condition does not by itself settle existence, since discontinuous games can have pure equilibria under weaker conditions (Reny 1999), but here a separate and specific argument closes the question: Any candidate pure profile is undone by profitable undercutting or overbidding, so no pure-strategy equilibrium exists.

1.1 A shared verification library

This paper presents `Econlib`, a Lean 4 library for formal economic and political theory built on `Mathlib` (Community 2019), the mathematics library of the Lean proof assistant (Moura and Ullrich 2021). `Econlib` is not a new result in the ordinary sense. It is research infrastructure in the form of a reusable collection of definitions, theorems, identities, and examples for economic and political theory. Its purpose is to provide a set of modeling assumptions, foundational results, and proof obligations to ease the formalization of research in economics and political science. Think of `Econlib` as playing the role that a statistical software package such as `R` or `Stata` plays for empirical work: A shared body of tools that users can rely on without rebuilding the underlying machinery each time.

A word on terminology is in order, because two research traditions use the word "formal" in different ways. In political science, "formal theory" refers to the use of mathematics—game theory, social choice, decision theory—to model politics (Austen-Smith and Banks 2000), as against verbal or statistical argument. In logic and computer science, "formalization" refers to rendering a definition, theorem, and proof in a language a machine can check. The two senses are independent: Most formal theory has never been formalized and most formalization is of mathematics no one would call formal theory. This paper is about formalizing formal theory by taking the deductive models that economists and political scientists already build by hand and rendering them so that a proof assistant can verify them. This paper uses *formalization* throughout in the machine-checking sense and names the modeling tradition *formal theory* or *deductive theory* where there is potential for ambiguity.

Formalization in the sense meant here means that the definitions, hypotheses, theorem statement, and proof steps of an argument are all written in a precise formal language checked by software, often called a "proof assistant". `Econlib` uses Lean, a proof assistant that is becoming the *de facto* standard for formalization in pure mathematics. Lean's kernel accepts a theorem only when it can verify that the claimed conclusion follows from the supplied definitions and assumptions. If a theorem in `Econlib` states that a voting rule is dictatorial under Arrow-style assumptions, that a VCG mechanism is dominant-strategy incentive compatible, or that one probability law dominates another in a stochastic order, the statement and its proof are machine-checked against hypotheses that must be explicitly provided. Assuming that the theorem has been correctly expressed in the formal language, and granting trust in the Lean kernel, one can be confident that the theorem is true. The first proviso is the critical one: A proof assistant can neither supply substantive judgment nor confirm that a named result like "the Second Welfare Theorem" is consistent with the textbook statement. Lean can only check that the definitions, assumptions, and proof steps actually support the claimed conclusion as encoded in the formal language.

Therefore the most important hurdle a formalization library must overcome is confidence in its faithfulness to the field it purports to formalize. To demonstrate the semantic correctness of `Econlib`, the main library is accompanied by two companion libraries. The first, `EconlibExamples`, contains worked examples of canonical results and models of interest to economists and political scientists and demonstrates that the expected results follow from the primitives in the main library. The second, `EconlibTest`, contains constructive witnesses (where

possible) and simple numerical checks to ensure that the library's main results are neither vacuous nor unfaithful to the concepts they claim to represent.

The contribution is therefore twofold. First, `Econlib` extends the reach of formal economics in Lean from isolated theorem formalizations to a reusable library that includes preferences, probability, optimization, game theory, mechanism design, social choice, and general equilibrium, with headline results derived from primitive definitions and no admitted proofs. Second, it contributes a form for future work: Domain-facing interfaces, type-level semantic constraints, carrier-specific probability interfaces, worked examples, and searchable documentation that let later modeling work import theory rather than rebuild it.

1.2 Roadmap

The paper first describes what `Econlib` formalizes at a high level and offers two brief concrete examples (the Second Welfare Theorem and existence of sequential equilibrium for finite signaling games) to show what a machine-checked economic result contains. With that concrete referent in place, Section 3 makes the case for formalization: It makes the scope of a claim legible from its statement, settles shared definitions so that results compose across papers, redistributes proof labor, and improves correctness. Section 4 then states the design conventions that make the library usable and composable, followed by a module-by-module tour in Section 5 that records each module's organizing design decisions and main results.

Section 6 situates `Econlib` in the literature on formal economics and political science and discusses related work. The remaining sections describe the intended path forward and the tools accompanying the main `Econlib` repository. The core claim made throughout is not that Lean automates theory but that a shared verification library can make hidden mathematical obligations explicit and give formal theory a reusable deductive infrastructure.

2 What `Econlib` Formalizes

2.1 A library organized around economic and political theory

`Econlib` is dedicated to the formalization of the economic and political theory that economists and formally-minded political scientists reason with and draw upon. Across preferences, probability, optimization, game theory, mechanism design, social choice, and general equilibrium, it defines the structures of each area—preference relations and their utility representations, probability laws and stochastic orders, games and their equilibrium concepts, mechanisms and transfers, voting rules, economies and Walrasian equilibria—and establishes foundational results about them. Along the way, `Econlib` has drawn from the enormous wealth of general mathematics available in `Mathlib`, the canonical collection of Lean-formalized mathematics, and has formalized pure-math results (such as Brouwer's fixed-point theorem) only as necessary for rigorous proofs

of the economics in the library. Results of interest to `Mathlib` will be gratefully upstreamed, but `Econlib`'s primary interest is in formalizing economic and political theory and providing an ergonomic interface for economists and political scientists. The results in mathematics are incidental to this task.

Readers familiar with `Mathlib` should note that, because the two libraries serve different users, they are organized along different axes. `Econlib` is arranged the way economic and political theory is used, by subject matter: Its top-level modules correspond to recognizable research areas, so a reader looking for the median voter theorem finds it under social choice, not under a general treatment of order or fixed points, and a reader looking for revenue equivalence finds it under mechanism design, beside the auction and screening machinery it depends on. Declarations are named and the library is organized for economists and political scientists rather than mathematicians, a point Section 4 develops, and similar mathematical topics reappear across `Econlib`'s subject modules when the same underlying machinery serves different economic objects.

What makes the library substantial is both its breadth and depth. `Econlib` covers its seven subject areas broadly (but cannot claim to cover them comprehensively) and proves all results from first principles with no admitted proofs or custom axioms. Definitions are stated in their most recognizable form and headline results are derived from primitive definitions all the way to their substantive conclusions, with every intermediate obligation discharged and the entire chain backed only by the typical Lean axioms.[3] `Econlib` also contains novel formalizations (as far as the author is aware) across the library: Strassen's theorem and Rothschild–Stiglitz mean-preserving spreads in probability; Debreu utility representation and Arrow–Pratt comparative risk aversion in preferences; Topkis/Milgrom–Shannon comparative statics and envelope results in optimization; Kuhn's theorem and sequential equilibrium in game theory; Myerson's lemma, revenue equivalence, ironing, and Myerson–Satterthwaite in mechanism design; Black's median-voter theorem in social choice; and Arrow–Debreu existence with the second welfare theorem in general equilibrium.

The "how" is as much the contribution as the "what." Rather than simply collect formal counterparts of familiar results, `Econlib` is designed with special care to make results reusable and composable. Its definitions are chosen so that results from one area can be imported by another without translation: Fixed-point arguments can support both game-theoretic and general-equilibrium existence theorems; stochastic orders can be used in probability, information design, and mechanism design; preference relations can be shared by welfare economics and social choice. The formal work is paid for once at the level where it is reusable and then exposed through the economic and political-science interfaces rather than raw topology, order theory, or measure theory. The sections that follow describe what is formalized in `Econlib` (this section and Section 5) and the project conventions that keep them usable (Section 4).

[3] That is, `propext`, `Quot.sound`, and `Classical.choice`.

2.2 Anatomy of a Result

To give a flavor of how `Econlib` is designed and the results it enables, it is more revealing to take a well-known result and peel it back to its primitives than to start from primitives and build up a theorem. Consider the Second Welfare Theorem (SWT), which states that any Pareto optimal allocation can be sustained, after lump-sum transfers, as a competitive equilibrium. In `Econlib` the exchange form is `exists_walrasianEquilibriumWithTransfers`, and it is stated as

```
theorem ParetoOptimal.exists_walrasianEquilibriumWithTransfers
  /-! # Requirements (hypotheses and necessary objects): -/
    {L : ℕ} (E : Economy L) - Some natural number `L` indexing commodities and an `Economy`.
    (hL : 0 < L) - There must be at least one commodity
    (hne : Nonempty E.Agents) - The set of agents must be nonempty
    (hreg : RegularEconomy E) - The economy `E` must satisfy `RegularEconomy`
    {x : E.Agents → Fin L → ℝ} - Candidate allocation: An agent's assigned bundle of goods.
    (hpareto : E.ParetoOptimal x) - The candidate allocation `x` is Pareto optimal
    (hcons : ∀ l, ∃ a, 0 < x a l) - Every commodity is consumed by some agent
    (hirr : Irreducible (E.transferEndow (fun a ⇒ hpareto.feasible.nonneg a)))
    /- `hirr`: The transfer relabeled economy is irreducible. -/ :
  /-! # Conclusion: -/
  ∃ W : E.WalrasianEquilibriumWithTransfers, W.Decentralizes x := by
  /-! Proof (See `Econlib/Equilibrium/SecondWelfare.lean`) -/
```

In Lean, the theorem statement is followed by a set of requirements that, when satisfied, yield the conclusion after the colon. Here, the SWT starts with an exchange economy `E`, requires a nonempty set of agents, regular preferences and endowments, at least one commodity, a Pareto-optimal candidate allocation `x`, the requirement that every commodity be consumed in a positive amount by some agent, and an irreducibility condition after endowments are relabeled by the transfers induced by `x`. From those inputs it concludes that there exists a Walrasian equilibrium with transfers that decentralizes `x`.

The value of this statement is that the conclusion is a structured object that guarantees the existence of the equilibrium. The theorem produces a `WalrasianEquilibriumWithTransfers`, a structure whose fields are the equilibrium conditions themselves. To build one is to supply prices that are nonnegative (`price_cone`) and not all zero (`price_ne`), transfers that sum to zero (`transfers_balance`), an allocation that maximizes each agent's preference within the transfer-adjusted budget (`isOptimal`), and a proof that every market clears (`clears`). A reader need not check that the object called an equilibrium really is one: An object failing any of these conditions is not constructible.

Notice also that the conclusion follows from hypotheses (the statements before the colon) that are themselves objects defined in the library. `Econlib`'s SWT applies only to a `RegularEconomy`, which is an `Economy`[4] that requires continuous, convex, monotone[5], desirable preferences for every agent and that every agent owns a nonzero endowment. It also requires the transfer-relabeled

[4]The `Economy` in this exposition is a minimal environment with finitely many agents, a preference over commodity bundles, and nonnegative endowments. The infrastructure beneath it is abstract over the agent type; the finite form is the exposed interface, a point the next subsection develops.

[5]The precise requirement is that preferences are strictly monotone toward interior bundles.

economy to be `Irreducible` so that there cannot be agents stranded with no wealth. These equilibrium-specific objects are discussed in Section 5.7.

The proof, omitted for space, also relies upon components built elsewhere in `Econlib` and follows the classical existence and welfare arguments of Arrow (1951), Debreu (1959), and McKenzie (1959). It first obtains supporting prices for the Pareto optimum, normalizes them to the price simplex, and uses positive aggregate consumption to show that some agent has positive wealth. It then treats `x` as the endowment profile in the relabeled economy `E.transferEndow ...` and invokes the library's quasi-equilibrium-to-Walrasian upgrade; irreducibility closes the zero-wealth gap that supporting prices alone leave open. The returned equilibrium already satisfies the equilibrium conditions by construction—prices, balanced transfers, optimality, and market clearing are its fields—so what the conclusion adds beyond bare existence is the separate predicate `W.Decentralizes x`: That this equilibrium implements the given Pareto optimum `x`, with each agent's transfer fixed to the canonical supporting value, the worth of its bundle minus the worth of its endowment at equilibrium prices. That `x` is the prescribed optimum, not merely some equilibrium allocation, is what makes the statement a second welfare theorem.

At the bottom of the dependency chain lies the preference relation (`PreferenceRel (Fin L → ℝ)`) carried by each agent. `Fin L → ℝ` defines the commodity space, a function mapping the commodity index set `Fin L`[6] to real quantities. `PreferenceRel α` is Econlib's type of preference relations over an arbitrary set of alternatives α, carrying a weak ranking $\succeq$ together with its induced strict preference and indifference and the proofs that it is complete and transitive. Instantiating α at the commodity space gives `PreferenceRel (Fin L → ℝ)`, which is a rational ranking of consumption bundles. The headline welfare theorem and the bare preference relation are the two ends of one verified mathematical chain.

The anatomy of the SWT definition and proof generalizes throughout the library. Beneath every Econlib result sit the same layers of basic primitives (like a preference relation or distribution) that support richer objects (like an economy or mechanism) that carry the constraints that make them economically or politically meaningful. Theorems are proved on these constrained objects using lemmas and identities—bridges, simplifications, transformations—that let a model pass between economic objects without rederiving the connection from scratch.

To take another brief illustration, `Econlib` proves the sequential-equilibrium existence result for finite signaling games, a workhorse of political economy from deterrence to electoral signaling. A signaling game is presented on the familiar surface of sender types, messages, receiver actions, priors, strategies, assessments, and off-path beliefs. The headline theorem `exists_signalingSequentialEquilibrium` states that, under a full-support prior, there is an assessment that is a Kreps–Wilson sequential equilibrium of the encoded extensive game. One layer down, the result rests on two bridges of the kind just described. First, `exists_signalingPBE` obtains a signaling perfect Bayesian equilibrium by solving perturbed signaling games and taking a limit assessment. Second, `isSignalingSequentialEquilibrium_of_isSignalingPBE` carries that equilibrium into a

[6] `Fin L` is defined simply as the set of natural numbers less than `L`, so the index set is $\{0, 1, \ldots, L-1\}$, one entry per commodity.

sequential equilibrium by turning signaling assessments into extensive-form assessments, Bayes-consistent beliefs into Kreps–Wilson consistent beliefs, and sender and receiver optimality into sequential rationality. The theorem also shows that the full-support hypothesis is necessary by proving that if some sender type has zero prior probability, no signaling sequential equilibrium exists.

2.3 Working With Econlib

As hinted at by the preceding discussion, theorem proving with Econlib is neither like defining a model and puzzling through its consequences as it might be when working with pen and paper nor like picking up Lean and working with the bare constructs Mathlib exposes.

Instead, one should expect to work in Lean but with the primitives exposed by the library; Econlib stands on Mathlib but moves the point of entry. The interface users are expected to work with has been designed carefully to be at a useful and recognizable level of abstraction. For example, the SWT proof discussed in Section 2.2 is presented on a finite economy, but the library infrastructure used internally is abstract over any agent type, not just finite agents.[7] Many objects in the library are pushed to broader levels of abstraction: `budgetSetAt` is stated at an arbitrary wealth level rather than only at endowment wealth, and `aggregateExcessOver` and `ParetoOptimalOver` are stated over an abstract aggregation functional rather than only over finite sums. But the default interface remains economic: One should expect to use `Economy`, `PreferenceRel`, `FinDist`, or `DirectMechanism`, not a generic record of feasible sets, objective relations, aggregation operators, and constraints. Among the design principles developed in Section 4 is to expose the most recognizable object that is still general enough to support reuse, so that users work with economic primitives while the reusable mathematics remains available underneath.

While Econlib is intended to be accessible to those without a background in formal theorem proving, using Econlib unavoidably rewards knowledge of Lean 4 and its tactic-driven proof writing. While a number of good pedagogical resources are available (such as Lean's official Theorem Proving in Lean 4 guide), the gallery of examples in `EconlibExamples` is extensively commented and should serve as a good domain-specific reference to using Lean and Econlib to formalize research in theoretical economics and formal political science. Each example is a self-contained, recognizable model—a second-price auction, a Condorcet cycle, a Cobb–Douglas Edgeworth box—proved using the full workflow, from stating the primitives to invoking the theorem necessary to reach the desired conclusion.

The next section makes the broader case for formalizing theory at all; the conventions that produce this economic interface then follow in Section 4, and the module-by-module tour in Section 5 presents each topic area, its main results, and its worked examples in turn.

[7] These internal objects are exposed as part of Econlib's API surface and so are available to use for those interested.

3 Why formalize?

Before spending more time discussing the design of the library, we should turn to the question of why formalizing deductive theory is a worthwhile exercise at all. The discipline of research mathematics, where proof assistants have had the most adoption and influence, has observed that formalization enables new ways of doing research and broadens the range of potential collaborators. Scholze, for instance, has emphasized that proof assistants allow mathematicians to offload working memory to the computer. Rather than requiring the researcher to maintain a detailed mental picture of the entire model being constructed, a proof assistant makes it possible to build proofs incrementally, with the assurance that the resulting pieces compose correctly and that no edge case has been silently overlooked. To the second point, Tao's survey of machine-assisted proof (Tao 2025) documents how proof assistants make large, loosely coordinated collaborations on technically complex material more feasible, because subtasks can be checked by a trusted oracle rather than accepted on the basis of a collaborator's reputation. As researchers increasingly turn to LLMs to build models and conduct research, this same checked infrastructure can serve as a guardrail against hallucination and imprecision. Tao anticipates precisely this division of labor, arguing that proof assistants and computer algebra systems can filter the now-notorious tendency of large language models to hallucinate, while LLMs can reciprocally automate the more tedious aspects of formalization (Tao 2025). Section 7.1 returns to this point.

The value of formalization is then not only (or even not chiefly) located in guaranteeing correctness. The consequences run along four lines, developed below: How precise definitions make a claim's requirements legible, how formalization redistributes the expertise a contribution demands, how verified results accumulate across a field, and finally what a verified theorem's correctness does and does not guarantee.

3.1 Formal definitions make scope legible

Good formal definitions clearly state the relevant objects and their ambient structure before the theorem is used or proved. A strategy does not simply "respect information sets" as an informal description but as an object built over an explicitly-defined information structure. Preferences are not left "well-behaved" in the background; completeness, transitivity, continuity, convexity, or single-peakedness appear as named structure or hypotheses.

This makes the theorem statement a compact account of scope. The statement says which obligations must be satisfied, which objects carry those obligations, and what environment the conclusion presupposes. A reader can see whether a model requires compactness, measurability, support, convexity, or equilibrium-consistency conditions without needing to work through the whole proof. The theorem is therefore easier to use and understand even when everyone agrees that the proof is correct.

Informal exposition often spreads these obligations across the paper. Some conditions are introduced in the setup, some are used only inside a proof, and some remain tacit because they are

standard in the leading examples. A formal development collects them in the definitions and theorem statements where scope is easiest to assess. Definitions become part of the argument because they say, before the proof begins, what environment must be in place for the claim to apply.

The cost is reorienting oneself to the theorems are stated in formal language, which is more like reading code than it is reading math in that one must trace dependencies and have a good grasp of the language's syntax. The SWT example in Section 2.2, for example, requires following `RegularEconomy`, `Irreducible`, and `Decentralizes` into their definitions and understanding that the argument to `Irreducible` builds the relabeled economy via an anonymous function. Formal languages also introduce repetitiveness and, depending on one's taste, irritating pedantry in that every routine hypothesis must be named even when it holds in all cases of interest. This is a cost inherent to formalization that is borne in part by the library, which states the standard assumptions once and passes them to every downstream theorem.

3.2 Shared definitions coordinate a field

Formalization also makes theory more cumulative in both technical and social senses. In the technical sense, formal proofs allow for more explicit modularization by describing interfaces and delineating abstraction boundaries as described by Commelin and Topaz (2023). In essence, this means that a formal statement should separate its *interface* (how it should be interacted with) from its *implementation details* (the proof path taken to establish the claim). This delineation is natural in a formal language and facilitates the use of results proved by others in one's own research. The analogy is to statistical software or empirical replication packages, which expose (for example) linear regression as an interface for researchers while abstracting away the underlying implementation such that it can be improved without impact on users.

In informal work, reusing a complex theorem often requires looking through the proof, reconstructing its technical assumptions, and deciding whether the same background machinery applies in the new setting. A verified declaration changes that division of labor. Once the result has been formalized, a later researcher can reuse its economic content directly: If a model satisfies the stated hypotheses, the theorem can be invoked without reopening the fixed-point argument, the measurability construction, the order-theoretic lemma, or whatever other machinery lies beneath it. The point is not that formalization makes judgment unnecessary but that it localizes judgment to the substantive questions of interest to the field. A verified library turns complicated analytic results into usable components: Definitions settled once, machinery imported rather than rederived, and proofs relied on without being redone.

The social sense follows as a corollary. Formal results, no matter their complexity, can be built upon regardless of one's trust in the competence of the author or one's familiarity with their work. As long as one applies their judgment as to whether the model is well-motivated and well-specified, trust in the details can be delegated to the formal language and the results repurposed with confidence.

Another benefit is that formalization fixes the meanings of definitions themselves. As successive models are built on the same primitives, the field accumulates not only theorems but a common vocabulary that cannot be used in slightly different ways in different papers and contexts. The slightly different meanings of perfect Bayesian equilibrium identified in the introduction of this paper, for example, are disambiguated in `Econlib`, which defines `IsPerfectBayesianEquilibrium` as sequential rationality together with Bayes consistency on the positive-probability information sets—the Fudenberg–Tirole convention, which leaves beliefs off the equilibrium path unconstrained—so that a model built on the library inherits exactly that notion. What a term means is thereby settled once rather than renegotiated paper by paper. Nothing about this benefit depends on anyone's proof being wrong or on any cost-benefit judgment about routine formalization; it addresses a coordination problem formal theorists already concede they have.

3.3 Formalization redistributes the required expertise

It might seem that formal verification narrows the set of people who can contribute to theory, since a result now has to be stated and checked in a manner most economists and political scientists are not familiar with. While it is undoubtedly true that formalization requires different expertise, the net effect is not that formalization makes theory more exclusive. Instead, formalization redistributes the expertise a contribution to theory demands in such a way that can in fact increase the accessibility of theory for both readers and writers.

The redistribution begins with how a model is built. Modeling on the frontier of deductive theory requires a familiarity with mathematics that, for many, was nurtured well before graduate school. Keeping track of proof obligations, preconditions, and potential points of failure requires a good mental model of the problem at hand and strong *mathematical* intuition for how a result might be proved. Writing a formalized model, on the other hand, can be done with the confidence that the mathematics must be correct if the theorem is accepted by the formal language. The consequence is that formalized modeling rewards domain expertise and political or economic intuition more than it does mathematical intuition. Researchers can then focus attention on the point it is they are trying to make while leaning on the proof checker to be confident in the intricate lower-level steps.[8]

That decomposition changes what a theorist has to know personally. In informal work, the same author often has to specify the substantive model, remember the relevant background mathematics, police side conditions, and judge whether the proof sketch is complete. In a formal development, those tasks can be separated. The theorist states the object and the substantive theorem; reusable library results supply standard mathematical machinery; small lemmas record model-specific steps; worked examples and searchable signatures help locate the right components.

[8]This point applies most cleanly to lemmas used internally by a main theorem. The main theorem still has to be semantically sound: Its hypotheses must be satisfiable and its conclusion must express the intended claim. Once that is true, auxiliary lemmas need not be audited as independent substantive claims. If their formal statements are sufficient for the main proof, the checker verifies both their proofs and their use.

The discipline becomes more teachable because it is organized around public routines rather than tacit proof-reading standards: Define the object, state the local obligation, search for existing results, and check that the hypotheses match.

The same modularity changes collaboration and readership. While researchers remain responsible for the content of their model and the faithfulness of its formalization, they may delegate bounded obligations to coauthors, students, formalizers, or language-model assistants with the confidence that Lean determines whether that contribution fits the surrounding proof. Similarly, readers still have to understand a theorem and its assumptions, but they do not have to understand the intricacies of a difficult proof to believe the claim. The point is not that formalization makes theory automatic—it manifestly does not—but that it allows researchers to move their scarce attention toward formulation and interpretation, while lower-level proof labor can be divided, reused, or delegated. Section 7.1 returns to the machine-assisted form of this division of labor.

3.4 A bounded correctness guarantee

Finally, formal verification brings the most obvious benefit that a proof is guaranteed to be correct, both in one's own contribution and in the theorems one invokes to prove their result. But this correctness guarantee is less intrinsically valuable than it might appear for two reasons. First, there is the simple observation that most proofs in the social sciences are correct, so the effort of formalization rarely yields the startling conclusion that a well-known result is not true as stated. Second, and more seriously, the Lean kernel verifies only that a theorem's conclusion follows logically from its stated hypotheses; it does not verify that the theorem formalizes the claim the author meant to make. A researcher can encode the wrong notion, prove a theorem about it, and mistakenly treat the result as a proof of the intended claim. Similarly, a theorem with inconsistent hypotheses may be formally true while saying nothing about any possible model, just as the statement "all unicorns have glittery horns" is true only because there are no unicorns.[9] The value of verification therefore depends on the care with which the theorem itself is stated, as Lean verification can only guarantee that the theorem's conclusion follows from its hypotheses as they are stated in the formal language.

Within those bounds, the guarantee is valuable because of where it leaves the burden. A machine-checked proof leaves the formal statement as the one thing a reader must carefully evaluate. Unlike a proof, which may be arbitrarily complex and mathematically unfamiliar, a theorem statement is nearly always written in the domain's own vocabulary. This lets a subject-matter expert vet (and then trust) the objects, hypotheses, and conclusion without needing to follow the intricacies of the argument that produced it. Bayesian persuasion, formalized in `Econlib`, is a good example of a result that is intuitive to theorists but difficult to prove from first principles without an unusually strong mathematical background. The conclusion is the understandable claim that the value of optimal persuasion is the concave closure of the sender's payoff over the receiver's beliefs, so disclosure helps whenever that payoff is not concave at the prior. The machinery necessary to

[9] Logicians would say that this proposition is *vacuously* true. The force of this is that one can also prove that "all unicorns have non-glittery horns"; when quantifying over an empty set, both `P` and `¬P` are true for its members

rigorously prove this claim is formidable, especially in the continuous case: Convex majorants, disintegration into regular conditional distributions, and Kantorovich–Rubinstein duality. In a formal treatment, the reader can confirm that the statement is the familiar Kamenica–Gentzkow result while the kernel vouches for everything that backs it.

The second source of value is structural. A well-chosen set of abstractions guards against accidental misstatement and vacuity by the researchers who invoke them. As described in the next section, `Econlib` seeks to provide these abstractions by creating building blocks for sophisticated theory that are carefully designed to easily compose in semantically-valid ways and avoid accidental misuse. Consider the example in the preceding section of the existence of a sequential equilibrium in a signaling game. Existence requires a full-support prior: If a sender type has zero prior probability, its information set is never reached, yet consistency still requires a belief there —forced to one at the type's own node, since the sender knows its type, while every trembled posterior there is zero—so the consistency conditions cannot be met. A development that dropped full support would be able to prove a vacuous result about a pathological game's sequential equilibria that holds only because, on any instance with a zero-prior type, the admissible belief systems form an empty set and there is no equilibrium left for it to constrain. In `Econlib`, because consistency is a property the equilibrium object must carry, the requirement appears when the result is stated rather than as something a reader must notice was missing (see also the discussion in Section 4.2). Notably, this guards against an error that has occurred in practice: The main theorem of Gottlieb and Zhang (2021) on long-term contracting with time-inconsistent agents quantified over the equilibria of a game in which, as a corrigendum later recorded, an equilibrium may fail to exist, so the theorem was correct but potentially vacuous as originally scoped (Citanna et al. 2023). A formal statement requiring a non-empty set of equilibria forces the existence question into view at the moment the theorem is stated.

4 Design principles

Meeting the ambitions of `Econlib` has required careful attention to creating a correct, ergonomic, and discoverable interface for those seeking to use this library in their research. To that end, Econlib applies a small set of conventions across its modules in order to present a familiar economic and political surface to users while defensively designing against the semantic drift that has been repeatedly warned against in this manuscript. Before turning to the module-by-module description in Section 5, this section presents the principles that informed their design.

A brief aside is warranted to reiterate the challenges faced by a formal verification library. The first and most obvious challenge is building the proofs, which is no trivial task. However, once the proof is complete and verified by the typechecker, one can be confident in its correctness and move on. The second and more insidious issue is the aforementioned semantic drift: A statement that claims to prove one thing in its naming and documentation but that in fact proves something else or that proves the intended claim vacuously. Overcoming this challenge is especially important for a library intending to provide reusable primitives for researchers to build on, since the validity

of consumer projects is only as good as the library constructs they are built from. The design principles discussed below are therefore concerned first with guaranteeing correctness and validity and, only after this has been satisfied, with elegance and usability.

4.1 Every result must be exercised

The first principle is that every result intended as part of `Econlib`'s public surface must be exercised by a consumer outside the file that proves it, either by a worked model in `EconlibExamples` or by a semantic witness in `EconlibTest`. These exercises are intended to guard against theorems that the kernel verifies but that are unusable because hypotheses are too strong, vacuous because the assumptions cannot be jointly satisfied, or misoriented because an inequality, transfer sign, or stochastic order was encoded backward. The example and test libraries make those failures visible at compile time and prevent regressions. This principle is the library's working answer to the faithfulness gap of Section 3.4: The kernel checks that proofs follow from statements, while the examples and witnesses check that the statements mean what their names claim.

Where possible, the consumer is a concrete model with numbers. For example, the Cobb–Douglas Edgeworth-box example fixes two agents, two goods, endowments `(2,1)` and `(1,2)`, price `(1,1)`, and allocation `(3/2,3/2)` for both agents, then proves demand optimality, exact market clearing, gains from trade, welfare, and uniqueness up to price scale. The corresponding equilibrium tests reuse the same economy to check the budget-set orientation, price homogeneity, Walras's law, and the exchange existence chain. The mechanism-design examples do the same for VCG: The two-bidder second-price auction imports `vcgMechanism_isDSIC`, then proves that the Clarke pivot is the rival's bid, a winner pays the second price, and a loser pays zero. These small models rule out broad classes of semantic mistakes.

For nonconstructive results, "exercise" means instantiating the theorem on a recognizable object and using the object it returns. The Second Welfare Theorem with transfers is tested on an off-equilibrium Pareto optimum in the Edgeworth box, `offAlloc 0 = (1,1)` and `offAlloc 1 = (2,2)`. The test discharges regularity, positive-consumption, and irreducibility hypotheses and obtains a `WalrasianEquilibriumWithTransfers` that decentralizes that allocation; a separate explicit construction determines the supporting price `(1,1)` and transfers `(-1,+1)`. The Strassen tests play the same role for probability, using the mean-preserving spread from δ_0 to $\frac{1}{2}\delta_{-1} + \frac{1}{2}\delta_1$ to check the martingale-coupling direction and convex-order orientation. A result is accepted and included in the library only when it survives both the proof and this kind of external use.

4.2 Invalid constructs should be impossible to express

The second principle is that an object's invariants are enforced by its type. Less succinctly, Econlib defines its objects so that one cannot be constructed at all unless the properties that make it meaningful are supplied with it. When a property is part of what an object means, it belongs in the object's type, not in a separate proposition the user must remember to assert.

To take a concrete example, consider the behavioral strategy of an extensive-form game. Its semantics require that a player act the same way at every history in one information set, since those histories are by definition the ones the player cannot tell apart. The natural and permissive encoding is to define a strategy as a function from histories to action distributions and rely on the user to guarantee that the function does not return one distribution at one history and a different one at another. This design admits objects that violate the semantics the game is meant to model by relying on the user to enforce its invariants.

Econlib instead enforces semantics by building the necessary constraints into the type. In this case, a `BehavioralStrategy` is indexed by the player's information sets. It is therefore impossible to express the malformed strategy stated above because the function would need a value at a history the domain does not distinguish. Information-set respect holds by construction before any theorem is stated.

The design `Econlib` rejects is to keep the permissive object and recover correctness with a side condition. This design would still define a strategy as a function on histories as in the naive case but add a predicate `RespectsInfoSets` that is carried as a hypothesis wherever it is needed. A behavioral strategy then need not necessarily be valid; the burden shifts onto consumers to validate that the function they are receiving is in fact a valid behavioral strategy. Supposing that the side condition is verified everywhere necessary, this is a fine design, but one that requires additional mental load (as one must track side conditions) and that is semantically misleading (as invariants may not hold when an object is encountered).[10]

4.3 Abstractions should be built for working theorists

The third principle is that the public surface should expose the object a theorist ordinarily writes down rather than the most general mathematical encoding available. `Econlib`'s public surface is built from preferences, payoffs, utilities, type profiles, strategies, budgets, allocations, prices, and transfers. More abstract representations remain available when a theorem needs them, but they are not the default objects a modeler must manipulate. This is the minimal-abstraction principle: Expose the least general object that still supports the intended reuse.

Probability illustrates the convention. Measure theory supplies a common representation for probability laws, but treating that representation as the entry point would obscure the calculations most are familiar with and introduce needless abstraction. `Econlib` therefore keeps finite, countable, continuous, mixed, and general probability laws on their native computational surfaces, while proving representation-independent results at the general level and transferring them back

[10]Both designs set aside here are taken, as a deliberate and documented layering choice, by the similar `EconCSLib` library of Bei et al. (2026) (Section 6.1): Its behavioral strategy is indexed by game states rather than information sets (the permissive encoding), with information-set respect deferred to a separate layer that carries the well-formedness conditions (`SameMoverOnInfo`, `SameActionsOnInfo`) as predicates rather than type-level constraints (the side-condition design). The choice is principled on their side: Their state-graph model admits infinite state spaces, which the strictly observation-indexed carrier in this library does not.

(the design is discussed in more detail in Section 5.2). The abstraction does its work beneath the surface without making the most general representation the user's ordinary interface.

4.4 Model objects are passed explicitly

Lean has a mechanism, called a typeclass, for filling in background structure automatically. This is appropriate when the structure is canonical as it often is in mathematics: The real numbers have their usual order; a measurable space supplies the sigma-algebra used by integration; a finite type often comes with the equality decisions needed to sum over it. In these cases the object is mathematical infrastructure, and asking Lean to find it automatically usually makes a theorem easier to read.

Economic and political primitives are different because there are very rarely canonical instances. For example, a set of alternatives does not determine one preference relation any more than a set of candidates determines one social choice rule. These objects are part of the model, not background structure. If Econlib asked Lean to infer them, the statement of a theorem would hide a substantive primitive, and failures would appear as technical elaboration errors rather than as ordinary modeling mistakes.

Econlib therefore represents noncanonical economic and political content as explicit structures: An `Economy` carries its agents, preferences, and endowments; a `DirectMechanism` carries its outcome and transfer rules; a social-choice profile carries the voters' preferences. The theorem statement shows which object is being used, and the proof state keeps it visible. Typeclasses remain in the supporting mathematics, where automatic inference supplies canonical structure, while the economic and political primitives remain named inputs to the model.

4.5 Results are discoverable

A library this size is usable only if someone can find the right theorem and recognize its hypotheses without already knowing where it lives. `Econlib` supports discovery along several paths, both in the repository and through hosted tools.

Inside the repository, a concept glossary maps economic and political terms and their synonyms to their declarations. This is especially useful because the library's naming conventions follow `Mathlib`'s, so declarations typically are named after their conclusions rather than by their familiar names (for example, the converse of Kuhn's theorem is named `mixed_realizes_behavioral` with its familiar name provided in the docstring). The example suite also aids discoverability: One of the easiest ways to find the API for a kind of model is to open the nearest worked example to your problem.

For interactive search, `Econlib` hosts the standard Lean tooling against its own build. A Loogle instance at lean4.party/loogle answers queries by type shape, so a user who knows the form of a lemma but not its name can search for it directly. A semantic index at lean4.party/search

answers natural-language queries. Rendered API documentation at lean4.party/docs presents every declaration with its signature and docstring, cross-linked by dependency.

These aids guard against accidental duplication, where a result already exists but the user does not find it and proves it again. The same discovery surfaces also matter for language-model-assisted Lean, a point Section 7.1 returns to. The aim, for either kind of user, is to locate a result, identify its assumptions, and see how to apply it, without already knowing its name or location.

5 `Econlib` by Topic Area

The design principles to this point are stated as cross-cutting rules. The library itself is organized by subfield, each briefly summarized in the subsequent subsections. `Econlib`'s modules delineate subfields and research areas in economics and political science; the hierarchy in Table 1 is schematic but sketches the intended structure. Each subsection records the module's organizing design decisions and main results rather than its full contents; the complete inventory of declarations and worked examples lives in the hosted documentation described in Section 10.

<table>
<tr><td colspan="4">Math</td></tr>
<tr><td colspan="2">Probability</td><td colspan="2">Preferences</td></tr>
<tr><td colspan="4">Optimization</td></tr>
<tr><td>MechanismDesign</td><td>Equilibrium</td><td>GameTheory</td><td>SocialChoice</td></tr>
</table>

Table 1: Schematic hierarchy of `Econlib` modules.

`Optimization` is shown as a full-width bridge layer because it draws on primitives from `Preferences` and `Probability` while supplying analytical machinery—argmax, maximum theorems, comparative statics, dynamic programming, duality, and optimal transport—to the field-facing modules below.

The supporting `Math` layer is foundational but is not usually a user-facing entry point, has no examples directory, and hosts results that may not be stated in a form suitable for use beyond their intended consumer.

5.1 Preferences

The `Preferences` module defines the primitive constructs used by consumer theory, social choice, general equilibrium, and comparative statics, and its position at the base of the hierarchy dictates its design: Declarations must be general enough to serve every consumer and weak enough that no consumer is forced to assume more than it needs. Central to the design is the choice of an ordinal preference relation, rather than a utility function, as the basic carrier, since many economic arguments are ordinal at their core even when theorists write them with utilities for convenience. The basic type is `PreferenceRel X`, a complete and transitive weak preference relation on an arbitrary outcome type `X`, so the same object ranks commodity bundles in an `Economy`, alterna-

tives in a voter-indexed social-choice profile, or policies on a line, and `Optimization.argmaxRel` can maximize it directly over a feasible set. Where downstream theorems need more structure, it is added through named, composable predicates rather than declared ad hoc—`RegularEconomy` requires `ConvexPreference`, Black's median-voter theorem requires `SinglePeakedRel`—giving the library a shared vocabulary for the minimal conditions under which a result holds.

Utility functions, the construct most researchers work with, are built on top in the canonical way. The predicate `RepresentsRealPreference R u` states that `u` represents the relation `R`; representation is immediate in the finite case, while the continuous case is Debreu's representation theorem, `ContinuousPreferenceRel.exists_continuous_utility_representation`. The reverse direction needs no hypotheses: `preferenceOfRealUtility` turns any real-valued utility into a `PreferenceRel`. Concrete functional forms (Cobb–Douglas, CARA, CRRA, Inada) are supplied as declarations bundled with their properties, so a utility tagged with a risk attitude inherits the consequences of that attitude—Jensen inequalities from concavity, Arrow–Pratt comparisons between agents, prudence and its precautionary-saving consequences from the third-derivative condition—rather than reproving them.

The worked examples fix recognizable models (a CARA agent facing a fifty-fifty bet, a symmetric Cobb–Douglas consumer, a three-voter electorate on a policy line) and derive the textbook conclusions using only the library's exported theorems, proving no new mathematics of their own. The median-voter example runs Black's theorem on the same `SinglePeakedRel` predicate the social-choice module reuses, one piece of vocabulary doing work in two topic libraries.

5.2 Probability

Probability, another foundational module, is where `Econlib` departs most tellingly from its roots in `Mathlib`. Whereas `Mathlib` defines a single probability primitive, `MeasureTheory.ProbabilityMeasure`, under which every theorem is stated at maximum generality, Econlib keeps a family of carriers, each retaining the computational form native to its kind: `FinDist` is a probability mass function whose expectation is a literal finite sum that Lean's tactics drive down to arithmetic, `CountDist` uses countable sums, `ContDist` a density and an integral, `MixedDist` finitely many atoms plus a density, and `ProbDist`, an alias of `Mathlib`'s `ProbabilityMeasure`, covers the general case. What keeps the family from fragmenting into disconnected theories is that every carrier embeds canonically into the general measure. Results that do not depend on representation—stochastic orders, supports, weak-$*$ limits—are proved once at the level of the general measure and inherited through the embedding, while the computational operations `expect`, `map`, and `bind` stay native to each carrier, each paired with a coherence lemma identifying it with its measure-theoretic counterpart. A user who builds a finite model in `FinDist` thus inherits the general stochastic-dominance and mean-preserving-spread theorems without leaving the finite-sum surface they started on.

The comparative vocabulary economists move among is given shared names and proved translations: `FOSD` connects to monotone expected utility, `SOSD` to concave monotone expectations, `MLRP` to both dominance and the single-crossing conclusions of comparative statics, and mean-

preserving spreads to the convex order and to Strassen-style martingale couplings, so screening, risk, and information-design arguments can each ask for the formulation they need. Bayesian updating is handled as a construction on laws rather than algebra on densities: The user supplies the facts that make conditioning legal (a nonnegative likelihood, integrability, a positive marginal for the observed signal) and the posterior re-enters the same carrier it came from. Finite Markov chains extend the design to dynamics, with stationary existence, ergodic convergence, stochastic monotonicity, and sample-path identities proved once on a transition object so recursive models can import their probabilistic bookkeeping instead of re-establishing it.

The named distribution families are packaged the same way. On the continuous side Gaussian, Beta, Gamma, exponential, and log-normal laws, and on the discrete side Bernoulli, binomial, Poisson, geometric, multinomial, and the Dirichlet and Dirichlet–multinomial laws, are each defined as a concrete instance of a carrier, so a family arrives already carrying its density or mass function, moments, and CDF and already connected to the general stochastic-order theory. Family-specific results—normal–normal signal extraction, the beta–binomial and Dirichlet–multinomial compounds, and the likelihood-ratio and convex-order comparisons within the Beta family—are then theorems in that same idiom rather than a separate catalog a model must re-derive.

Among the worked examples, `MontyHall` walks the full design, building a prior and likelihood, deriving the posterior, and bridging the finite calculation to the general `ProbDist` law to show it is the finite face of one probability theory rather than a parallel one. `FirstOrderDominance`, `RothschildStiglitz`, `EmploymentChain`, and `GaussianSignal` each turn a distributional assumption into an economic conclusion on its native carrier.

5.3 Optimization

`Optimization` is the library's choice engine, turning "an agent optimizes" into reusable theorems for the field-facing modules. Its first design decision is ordinal optimization. Alongside the real-valued `argmax` and `valueFunction`, the module exposes `argmaxRel`, which maximizes a `PreferenceRel` directly by returning the greatest elements of a feasible set under the order, so conclusions that are order-theoretic—existence of maximal elements on finite spaces, uniqueness of the maximizer under strict convexity, relation-level Milgrom–Shannon comparative statics—are proved on the order alone, with no cardinal scale introduced. An agent's Walrasian demand in the equilibrium module is literally the `argmaxRel` of its preference over the budget set. The one property of choice that is not order-theoretic is the topological behavior of the maximizer correspondence, and there the module changes representation deliberately: Berge's maximum theorem is proved for the real-valued `argmax`, and a model that needs continuous demand crosses over through `argmax_eq_argmaxRel_of_represents`, with Debreu's representation theorem supplying the continuous utility. Everything ordinal is settled ordinally, and a cardinal representation is summoned only where necessary.

The second decision is to consume, rather than reprove, the analytic substrate. In constrained optimization, KKT sufficiency is drawn from the `Optlib` convex-optimization library (Li et al.

2024),[11] while `Econlib` contributes the recognizable economic layer: The saddle-point-to-maximum algebra with the expected sign conventions, and the abstractions that assemble a verified global optimum from first-order data, concavity, Slater's condition, and strong duality. The same economy of proof governs the envelope theorems, where Hotelling's lemma, Shephard's lemma, Roy's identity, and the Benveniste–Scheinkman formula are four corollaries of one Danskin-type theorem on the subdifferential of a family of maxima, and optimal transport, where Kantorovich–Rubinstein duality for the Wasserstein-1 problem is proved here for the persuasion module to consume.

Dynamic programming rests on a single contraction theorem: The Bellman operator is shown to be a contraction under Blackwell-style conditions once, then value iteration, optimal-policy selection, and the principle of optimality are built on the resulting fixed point. The dynamic-programming types are layered rather than parallel—deterministic, finite-state, and stochastic cores, a weighted-norm layer that admits the unbounded logarithmic and CRRA rewards a bounded theory cannot reach, and a collateral-specific layer for constrained-savings models—mirroring the carrier design in probability of a shared core with opt-in refinements, each adding exactly the structure its problem class requires.

The cleanest example is `BudgetMaximumTheorem`, which shows that the four conditions Berge's theorem requires follow from the budget-correspondence lemmas and that continuity of indirect utility plus upper hemicontinuity of demand follow with nothing interposed. `DiscreteCakeEating` obtains the value function and optimal policy from a short list of economic obligations and `CollateralLogUtility` shows the weighted-norm layer collapsing an unbounded-reward problem to a single boundedness certificate, in place of the hand-built transversality arguments its sibling `OptimalGrowth` assembles.

5.4 Game Theory

Game theory is where `Econlib`'s commitment to shared primitives is most visible because the subject most invites duplication. Deviation-based solution concepts share a logical core—fix the relevant notion of feasible unilateral deviation and require that no such deviation be profitable—and if each concept is formalized from scratch, a library can look faithful to the textbook taxonomy while admitting subtle variation in the deviations its concepts quantify over. `Econlib` therefore factors the common part into `EquilibriumProblem`, which is a structure containing a strategy space, a type of deviators, a relation giving each deviator its legal alternatives, and a value function, with `IsEquilibrium` requiring that no deviator can move to a legal alternative with higher value. Pure and mixed Nash in strategic games, Bayes–Nash for player–type deviators under interim expected payoff, and correlated equilibrium for player–recommendation deviators are specializations of the one notion, so ordinary statements keep their familiar form while the

[11] Econlib depends upon a small portion of `Optlib`. Its convex-function theory supplies the sufficiency half of the Karush–Kuhn–Tucker conditions in constrained optimization. Its subgradient calculus underpins the Danskin theorem from which the envelope results (Hotelling, Shephard, Roy, Benveniste–Scheinkman) follow. And its Farkas lemma backs the separating-hyperplane arguments that the equilibrium module's existence and welfare proofs consume.

formal definitions remember exactly which coordinate may change and which expectations are taken.

The shared spine pays twice over. First, it makes translation between solution concepts precise: A `GameMorphism` packages a map on strategies, a value-preservation condition, and a backward lifting of target deviations to source deviations, and `GameMorphism.transport` then carries equilibria across, so that a mixed Nash equilibrium induces a correlated equilibrium and a Bayesian game unfolds into its expanded strategic form without *ad hoc* arguments. Second, it organizes existence: `NashExistenceData` packages the compact convex strategy slices, continuous payoffs, and best-response structure needed for Kakutani-type fixed points. Finite mixed Nash, symmetric equilibrium, evolutionary stability, and finite and measurable Bayesian existence all reuse `NashExistenceData.exists_equilibrium` rather than repeating the fixed-point proof.

The extensive-form layer applies the design rule of Section 4.2. A `BehavioralStrategy` is indexed by player observations, not by arbitrary histories, so a strategy that conditions differently on two histories in one information set cannot be written down. The familiar refinements (subgame perfection, perfect Bayesian equilibrium, sequential equilibrium, belief consistency, perfect recall, Kuhn equivalence, one-shot deviation) sit on a representation in which the information constraints are already part of the strategy and belief objects. Signaling games, a workhorse of political science from crisis bargaining to electoral competition, get a modeler-facing surface—sender types, messages, receiver actions, priors, assessments, pooling and separating behavior, off-path beliefs, and the Cho–Kreps intuitive criterion—that `Econlib` bridges into the extensive-form framework, which is how the sequential-equilibrium existence result of Section 2.2 is assembled. The cooperative layer departs from the deviation spine deliberately, defining transferable-utility games, the core, balancedness, Harsanyi dividends, and the Shapley value on the primitives cooperative questions actually use rather than forcing the interface through coalition deviations.

The examples exercise every layer on canonical models: `PrisonersDilemma` and `MatchingPennies` in strategic form, `Rubinstein`, `Stackelberg`, and `EntryDeterrence` for backward induction on trees, `BeerQuiche` and `SpenceSignaling` for off-path beliefs and refinements as first-class objects, `CorrelatedCournot` for a linear Bayes–Nash equilibrium on a correlated Gaussian type space, and `MajorityGame` for the cooperative contrast between a symmetric Shapley value and an empty core.

5.5 Mechanism Design

Mechanism design covers two questions with different primitives. In a transfer problem the designer chooses allocations and payments and agents respond by reporting types or sending messages; in an information-design problem the designer chooses what agents learn and agents respond by updating beliefs. There is no representation that subsumes both, so the transfer surface builds on the game theory module while persuasion speaks the language of probability, convex order, concavification, and optimal transport.

On the transfer surface, an `IndirectMechanism` (a message space, an outcome rule, and a transfer schedule) induces a finite Bayesian game whose Bayes–Nash equilibria are the mechanism's equilibria, restated by `IndirectMechanism.IsBNE_iff` as the familiar interim no-deviation condition. A `DirectMechanism` restricts messages to type reports, and the revelation principle is then a theorem on these objects: `directify` builds the direct mechanism that reports types and plays a given equilibrium on the agents' behalf and `directify_isBIC` proves that the result is Bayesian incentive compatible, so the restriction to direct mechanisms is without loss of generality. The quasilinear layer covers interim and ex-post utility, Bayesian and dominant-strategy incentive compatibility, individual rationality, no-deficit conditions, the taxation principle, and the Groves and VCG mechanisms with their Clarke-pivot, efficiency, and participation properties. A single-parameter layer adds the one-dimensional structure of Myerson mechanisms—monotonicity, the envelope payment formula, Myerson's lemma, revenue equivalence, virtual values, regular optimal auctions, and ironing—specialized in an auction sublayer to symmetric independent private values, where the first-price bid schedule is a Bayes–Nash equilibrium in the measurable Bayesian-game sense, and carried by the bilateral-trade layer to the Myerson–Satterthwaite impossibility.

On the information-design side, the finite layer encodes the standard Kamenica–Gentzkow notions of Bayesian persuasion (Kamenica and Gentzkow 2011), with a `SignalStructure` inducing posteriors, `BayesPlausible` recording the plausibility constraint, `concaveClosure` giving the value, and `caratheodory_simplex` bounding the number of signals a finite problem needs. A continuous encoding follows Dworczak–Kolotilin convex-order duality (Dworczak and Kolotilin 2024) through weak duality, primal and dual attainment, complementary slackness, and KR-Lipschitz strong duality. The mathematics these rest on—convex order and martingale couplings, concavification, Kantorovich–Rubinstein duality—comes from the probability and optimization modules rather than being rebuilt as information-design lemmas.

The examples run across both surfaces, from the `ProsecutorJudge` persuasion benchmark and the continuous `GradeInflation` disclosure case through the `SecondPriceAuction` and `PublicGoodProvision` exercises of the VCG API to the regular, irregular, and bilateral single-parameter problems in `UniformRevenueEquivalence`, `MyersonIroning`, and `MyersonSatterthwaiteUniform`.

5.6 Social Choice

Social choice is the part of the library where economics and political science meet most directly. The subject is a frequent target of (and a useful stress test for) formalization because its best-known results are sensitive to small modeling choices. Arrow's theorem, Gibbard–Satterthwaite, May's theorem, and Black's median-voter theorem each depend on a different combination of domain, output object, and behavioral or normative requirement. Stating all four over the same primitives makes that sensitivity legible. The primitive is a `Profile Voter Alt` assigning each voter a `PreferenceRel Alt`, so domain restrictions (universal, strict, single-peaked) are hypotheses about the profiles a rule may face; what turns Arrow's impossibility into Black's possibility is exactly such a restriction.

The module separates a `WelfareFunction`, which returns a social ordering, from a `ChoiceFunction`, which returns a nonempty winner set, because the distinction changes the theorem: Arrow aggregates rankings under Pareto and independence of irrelevant alternatives and is stated over the former, while Gibbard–Satterthwaite concerns resolute choice under strategy-proofness and is stated over the latter, so each keeps the hypotheses it has in the literature. The standard assumptions—Pareto conditions, IIA, dictatorship, anonymity, neutrality, positive responsiveness, Condorcet criteria, strategy-proofness—are named predicates shared across results, so the dictatorship that appears in Arrow is definitionally the one that appears in Gibbard–Satterthwaite. Where a rule may return ties, strategy-proofness is not well defined until one says how voters rank winner sets; `Econlib` records the optimistic, pessimistic, and Kelly variants, proves the implications that hold among them, and separates the rest with finite examples, so a strategy-proofness theorem asserts something different under each.

The main results occupy recognizable roles. Arrow carries the decisive-coalition argument through universal- and strict-domain variants; Gibbard–Satterthwaite builds a welfare function from a strategy-proof resolute choice function, applies strict-domain Arrow, and transports the dictator back; May characterizes majority rule on two alternatives; Black gives a Condorcet winner on single-peaked profiles. The controlled profile transformations these proofs depend on (move-to-top, pointwise updates, pair and triple lifts) are formal operations rather than prose devices, and the named procedures (majority, Borda, plurality, scoring rules) are defined as welfare or choice functions, so failures of IIA, Condorcet consistency, or spoiler resistance appear as theorem-level facts about the same objects. The examples keep the finite intuition visible, from the `CondorcetParadox` cycle through the `BordaPathologies` and `PluralitySpoiler` failures to an explicit manipulation of resolute Borda in `GibbardSatterthwaiteManipulable`.

5.7 Equilibrium

The central theorems of general equilibrium rest on the library's heavier imported mathematics—existence on fixed-point theorems, the two welfare theorems on separating-hyperplane arguments—so the module is a direct test of whether that machinery can be reached through an economic interface rather than reproved in place. An `Economy L` has agents, nonnegative endowments, and for each agent a `PreferenceRel (Fin L → ℝ)` over commodity bundles, and the formulation keeps the economic content ordinal for as long as possible: Demand is `Optimization.argmaxRel` of the preference over the budget set, and feasibility, Pareto optimality, the core, and market clearing are stated over allocations and preferences. Utility representations enter only where a theorem needs the topological regularity they provide, chiefly the existence proof.

A `WalrasianEquilibrium` spells out what competitive equilibrium requires: nonnegative nonzero prices, allocations lying in each agent's demand correspondence, and free-goods market clearing, with Walras's law and the price-positivity lemma alongside. Existence has the familiar Arrow–Debreu shape: For a `RegularEconomy` (continuous, convex, desirable, monotone preferences and nonzero endowments), Kakutani's theorem yields a quasi-equilibrium on a truncated price simplex, and McKenzie-style irreducibility upgrades it to a Walrasian equilibrium with

strictly positive prices. The first welfare theorem places every equilibrium allocation in the Pareto set and the core. The second, as Section 2.2 showed, runs the other direction and asks strictly more: Without the consumed-good and irreducibility hypotheses a Pareto optimum need not decentralize at all. Production extends the same questions to economies with firms, where a `ProductionEconomy` adds technologies and ownership shares so that wealth includes profit income, and the welfare, transfer, and existence results carry over under the production irreducibility condition `IrreducibleProd`, whose substance the `FirmConnected` example isolates: Pure exchange irreducibility fails there, but production links the economy enough to recover existence.

Supporting files connect the abstract theorems to familiar calculations—closed-form Cobb–Douglas demand, Roy's identity against KKT multipliers, and the stationary accounting behind recursive competitive-equilibrium examples—and the examples keep the general results close to standard models: `CobbDouglasEdgeworth` proves equilibrium, welfare, and uniqueness in a two-agent Edgeworth box, while `FiniteExistence`, `RobinsonCrusoe`, `CobbDouglasRoy`, and `MarkovStationary` instantiate existence, production, duality, and stationary-equilibrium machinery in turn.

5.8 Supporting Mathematics

Underlying the results in the economic modules described in the preceding sections is the `Math` module, which proves the pure mathematics results necessary for results elsewhere in the library. Unlike the other modules, `Math` is not intended to be an entry point for users and so is neither directly tested (though all statements are expected to be tested transitively by its economic consumer) nor endowed with a suite of examples. `Math` is also prohibited from importing from other modules in Econlib. The dependency graph therefore points in one direction: Math exists to support the economic interfaces.

This supporting role is played in concert with `Econlib`'s dependencies (`Mathlib` and `Optlib`). Where possible, `Econlib` prefers to import ready-made theorems from its upstream libraries rather than prove specializations in `Math`, and not every local result is appropriate for upstream contribution. Many declarations in `Math` are cut to the shape of a single economic consumer, such as the affine-plus-hinge interpolation built for the Rothschild–Stiglitz duality or the totalized stop-loss transform suited to the convex-order characterization that consumes it, while more general results that fill gaps in `Mathlib` (compactness of the convex hull of a compact set in finite dimensions, the Jankov–von Neumann uniformization of analytic relations, Stieltjes integration by parts for monotone functions) are candidates to upstream. The layer also carries the fixed-point theorems the equilibrium and game-theory existence proofs rest on: Brouwer, proved from a cubical Sperner's lemma over a Freudenthal triangulation, and Kakutani, obtained from Brouwer through Caratheodory's theorem and a partition of unity. This is the classical route, shared with an independent Lean 4 formalization by harfe (2025), from which `Econlib`'s development differs in its combinatorial core (a `ZMod`-valued parity induction over the triangulation rather than a graph degree-sum count).

6 Prior Work in Formal Economics and Political Science

Formalization in economics and political science has developed in two strands. The older strand consists mainly of single-result formalizations and computer-assisted searches for impossibility theorems. These projects show that proof assistants and solvers can check important economic arguments, expose hidden hypotheses, and make textbook disagreements precise. They generally do not aim to provide reusable, domain-facing APIs. The newer strand consists of concurrent Lean projects released in 2026, including the two EconCSLib projects discussed below. They make the relevant comparison one of scope and orientation rather than one of library versus non-library work.

Social choice drew early attention because its central results are short to state and unusually sensitive to their hypotheses. Arrow's impossibility theorem has been formalized in Mizar Wiedijk (2009) and Isabelle/HOL, where Nipkow (2009) formalizes two proofs of Arrow, derives Gibbard–Satterthwaite as a corollary, and records gaps in the informal literature that the mechanization exposed. Grandi and Endriss (2009) recast Arrow as the non-existence of a finite model of a first-order theory, the reduction the automated-reasoning work then exploits. Among the earlier projects, the closest to a reusable framework is Holliday, Norman, and Pacuit (2021), a Lean development for voting theory over variable electorates that formalizes axioms such as Condorcet consistency and independence of clones rather than a single impossibility.

A parallel line uses satisfiability solvers to find and certify social-choice results. Tang and Lin (2009) reduce Arrow, Gibbard–Satterthwaite, and Sen's theorem to finite base cases a computer can check; Geist and Endriss (2011) search a space of preference-extension axioms and return dozens of impossibilities, known and new; and Brandl et al. (2018) establish an incompatibility for randomized rules with an SMT solver and re-verify the minimal unsatisfiable core in Isabelle/HOL. This work aims at discovery and certification. The proof assistant enters mainly as a final check on the solver rather than as a place to accumulate reusable objects.

Game theory, mechanism design, and general equilibrium have each been formalized in single-result form. Vestergaard (2006) gives a constructive Coq proof that finite sequential games have Nash equilibria, and Le Roux (2009) characterizes the preference condition under which they do; Parsert and Kaliszyk (2018) formalize the von Neumann–Morgenstern expected-utility theorem in Isabelle/HOL. In Lean, the earlier game-theory development at math-xmum (2023) formalizes von Neumann's minimax theorem along with Nash, Sperner, and auction-theory machinery, and predates the EconCSLib line discussed in Section 6.1. Mechanism design has a literature centered on auctions: The ForMaRE project Lange, Rowat, and Kerber (2013) and Caminati et al. (2015) verify Vickrey and VCG auctions in Isabelle/HOL and extract executable, verified auction code, while Barthe et al. (2015) give the first formal proof of incentive compatibility for VCG. For general equilibrium, Kaliszyk and Parsert (2018) formalize a pure exchange economy and an Arrow–Debreu private-ownership economy in Isabelle/HOL and prove the first welfare theorem for both, reconciling textbook definitions that disagree. Work has recently begun on macroeconomics, with a formalization of the Solow growth model Koepke and Schäfer (2026), and

on market design, with a verified continuous-double-auction matcher in Coq Garg and Sarswat (2022).

Two features separate this earlier body of work from `Econlib`. The first is reach. `Econlib` carries general equilibrium past the first welfare theorem to existence, through Kakutani and irreducibility, and on to the second welfare theorem and production economies, the parts that need the imported topology. The second is form. Most earlier developments are organized around a theorem or a solver pipeline; `Econlib` is organized around objects, and the artifact is a vocabulary the next model imports. Before the 2026 Lean economics libraries surveyed in the next section, the closest precedents for that form lay mostly outside economics, in `Mathlib`-style domain libraries such as the Lean computer-science library Barrett et al. (2026). `Econlib` brings that model to economic and political theory. The earlier attempt to bring mechanized reasoning to economists rather than to formal-methods researchers, Kerber, Lange, and Rowat (2016), surveys the social-choice and auction formalizations above and shares `Econlib`'s audience, but predates the library infrastructure that makes the agenda practical.

The enabling backdrop is the formal turn in mathematics itself Avigad (2023), where collaborative developments such as the Liquid Tensor Experiment Scholze (2021) showed that current proof assistants carry research-level mathematics, and where the practice of separating an interface from its proof Commelin and Topaz (2023) is what lets a library compose. `Econlib` applies that practice to economic and political theory.

6.1 Concurrent libraries for economics in Lean

Two projects released in 2026 while `Econlib` was under development share its aim of a reusable library rather than a single proof. Both projects use the name `EconCSLib`; this subsection labels them by author, Bei et al. and Garg, to disambiguate between them. Both are built on Lean and `Mathlib`. They were identified only as this paper was being written and are independent, concurrent work.

The closest in spirit to `Econlib` is the `EconCSLib` of Bei et al. (2026), which sets out to play for computational economics the role `Mathlib` plays for mathematics. Parts of `Econlib`'s and `EconCSLib`'s designs has converged, such as a representation of a strategic game that carries strategy spaces and payoffs with deviation as a pointwise update. That two groups working without knowledge of each other reached the same conventions provides a signal of correctness for each.

A closer look shows those shared conventions resting on different foundational choices which propagate throughout their design. The most consequential difference is easiest to state in terms of what each library can reach. `Econlib` covers continuous-type mechanism design (revenue equivalence, ironing), Bayesian persuasion with its measure-theoretic duality, and general-equilibrium existence through real topology, results that `EconCSLib`'s design cannot express; `EconCSLib` in turn supports exact arithmetic and executable, machine-checkable verification procedures that `Econlib`'s design cannot produce. Both trajectories trace back to one decision about the carrier:

Bei et al. (2026) keep payoffs and values polymorphic over an ordered type, which enables `decide`-style computation and verified checkers, while the present library fixes them to the reals, which buys `Mathlib`'s measure theory, convexity, and real analysis, the substrate the continuous-type and equilibrium results above require. The reuse strategies differ in kind as well. `EconCSLib` tends to give each solution concept its own predicate and pair it with a checker; `Econlib` routes whole families of concepts through a few shared abstractions—a single `EquilibriumProblem` behind the Nash-style concepts, one `NashExistenceData` existence engine, a `GrovesData` family behind the mechanisms. Even where the two encode the same object the index can differ: `Econlib`'s make-invalid-states-unrepresentable principle (see Section 4.2) leads it to index a behavioral strategy by the player's observations, whereas `EconCSLib` indexes by game state and recovers information-set respect through a separate layer. None of these are disagreements about what is true; they are different priorities about what the encoding should make easy.

The libraries also face different subjects. `EconCSLib` faces computation and computer science, with fair division, matching, online algorithms, fixed-point, minimax, and linear-programming machinery, and open problems posed as machine-checked benchmarks for automated provers. `Econlib` faces economic theory and formal political theory, with probability carriers and stochastic orders, continuous-type mechanism design with revenue equivalence and ironing, Bayesian persuasion and information-design duality, dynamic programming, equilibrium refinements, and the general-equilibrium existence and welfare theorems. Where the two overlap—on strategic, extensive, and cooperative games, on Arrow and Gibbard–Satterthwaite, on the core and the Shapley value, and on Vickrey, VCG, and Myerson's lemma—they are best read as corroboration; where they diverge, as complementary maps of the same terrain.

The second project of the same name, by Garg (2026), is organized differently. Rather than a concept library, it is a workflow for formalizing individual research papers with a language model and promoting reusable lemmas to shared infrastructure as they arise. It is complementary to `Econlib`. A paper-formalization pipeline of that kind is a natural consumer of the foundations a library supplies, a point Section 7.1 returns to.

7 Using `Econlib` in your research

Formalization adds friction to model building. A result that takes an afternoon to sketch on paper can take days to state and prove in Lean and using the library rewards a fluency in Lean and its tactics that most economists and political scientists do not yet have. Section 3.3 argued that this cost falls unevenly in a useful way and Section 7.1 explains why language-model assistance is reducing it, but the cost still has to be paid. `Econlib` is therefore unlikely to make routine formalization the default for every new theoretical result. Its practical value is at the margin where the cost of checking is lower than the cost of leaving a mathematical obligation implicit.

That margin appears most often in three cases. The first is uncertainty about a proof. If a result depends on a delicate argument that one is not sure of, formalization can locate whether a hypothesis is missing and help repair the claim. The second (and related) is unfamiliar mathe-

matics. A theorist using convex analysis, stochastic orders, optimal transport, or measurable selection can import a checked statement rather than rely on a remembered theorem that may not apply to the objects in the model. The third is reuse. A result that will become a lemma for later papers, an appendix result for a disputed proof, or a classroom example is a better candidate for checking than a one-off comparative static whose assumptions and proof are already transparent.

The research workflow need not begin with a full formal counterpart to a paper. A useful formal artifact can be narrower: A checked version of the main theorem, a formal appendix for the hardest lemma, or a model file that records the primitives and verifies that the canonical theorem being invoked really applies. In practice, the starting point is usually the nearest worked example in `EconlibExamples`, the relevant signatures in the generated documentation, and the theorem statement one wants Lean to accept. The formal file then becomes analogous to a replication package. It does not replace the paper's exposition, but it gives readers and later authors a checked object to inspect, extend, and import.

7.1 `Econlib` and LLM assistance

Language-model assistance changes the cost of this workflow without changing the verification standard. A model can draft definitions, search for plausible lemmas, fill routine tactic proofs, and explain failed proof states, but the Lean kernel remains the authority on whether the resulting argument checks. This makes the combination useful in exactly the setting where ordinary LLM use is most fragile. While the researcher still has to decide whether the formal statement expresses the intended economic or political claim, the lower-level proof labor is exposed to a mechanical verifier rather than accepted from the assistant's prose. Hallucinations or invalid proof steps are rejected by a deterministic oracle.

`Econlib` supports this use directly by shipping an optional agent skill, following the agent skills convention, for use with coding agents such as Codex or Claude Code. The skill packages the repository-specific habits that make an assistant useful rather than merely fluent in Lean: Search the glossary and codebase before introducing a new declaration; start from the closest worked example; use the hosted Loogle, semantic search, and documentation surfaces; run `lake env lean` or the appropriate `lake build` target after edits; avoid duplicating `Mathlib` or `Econlib` API; and preserve the design rule that semantic constraints should live in the objects being constructed. These instructions matter because the hard part of library use is often discovery and fit, not syntax alone.

The longer-horizon possibility is autoformalization: Taking an existing theory paper and producing a checked counterpart, with the machine carrying more of the mechanical labor and the researcher supplying judgment about whether the formal statement is the intended one. The enabling tools are advancing quickly in general mathematics. Lin et al. (2025) trains an open-source model that autoformalizes natural-language mathematics into Lean 4 at the scale of a million statements and then proves them, reaching state-of-the-art open results on standard proof benchmarks. The Hilbert agent of Varambally et al. (2025) orchestrates an informal-reasoning model, a Lean prover, and the kernel together, decomposing a goal recursively and repairing

failed proofs against verifier feedback. Workflows aimed at economics are taking shape on the same footing: Garg (2026) formalizes economics-and-computation papers one at a time with a language model, promoting reusable lemmas into shared infrastructure as they arise. A library like `Econlib` is the foundation such a pipeline draws on. The division of labor is the one Section 3.3 described, with the machine taking a larger share of the proof and none of the interpretation.

8 The Road Ahead

The most immediate work is breadth. `Econlib`'s topic coverage reflects the interests of its maintainer, so important areas of economic and political theory remain thin or absent. The next planned expansion is a statistical inference and econometrics module, so that formal models can be connected more naturally to the assumptions used in empirical designs. Contributions in this and other areas are welcome when they fit the subject areas of the library and follow the design conventions in Section 4: Recognizable domain objects at the public boundary, semantic constraints encoded in the relevant types, reusable results stated at the level later researchers will want to import, and accompanying examples and tests to demonstrate semantic validity.

A second task is stabilization. Pre-1.0 development favors getting the abstractions right over preserving every name, but a usable research library eventually needs a dependable public surface. That means clearer separation between internal machinery and supported API, more examples that show how to combine modules, and documentation that makes theorem discovery less dependent on knowing the library's file structure. The examples and generated signatures already serve this purpose; the next step is to make them comprehensive enough that a new user can begin from a model they recognize.

The larger question is institutional. A replication package lets an empirical claim be re-run; a checked artifact lets a theoretical claim be re-derived. It is not yet clear which theoretical claims, journals, subfields, or collaborations will come to expect such artifacts. What a library controls is the cost side of that question, by making the checked artifact something a later author assembles from importable parts rather than builds from nothing.

9 Conclusion

Deductive theory, in economics and political science alike, is the part of these fields the credibility revolution passed over, even though it is the part most open to mechanical checking. A theorem rests only on definitions and deductive steps, yet it is still certified by prose, referees, and trust, and the obligations that decide whether its conclusion holds are scattered through the argument and easy to lose.

Econlib gives theory the kind of shared artifact empirical work built for itself. It is a machine-checked library of economic and political objects and results, shaped for economists and political scientists rather than for the mathematics underneath, in which the hypotheses a result needs are

written into the objects that carry them and a later model imports what earlier work established instead of reproving it. The guarantee is bounded, the coverage is partial, and the cost of use is real. What changes is the unit a theorist builds from. As more models rest on the same definitions, the field accumulates not only theorems but a settled vocabulary, and a correction to a foundational result reaches everyone who depends on it. That is the sense in which a verified library makes theory cumulative, and it is why the infrastructure is worth building even though no single theorem in it is new.

10 Availability

Econlib is versioned as `0.29.0` in its Lake package file and builds against Lean 4 `v4.29.0` and `Mathlib` `v4.29.0`. The main library contains about 500 Lean files across eight top-level modules and is sorry-free: It contains no admitted proofs. The companion libraries `EconlibExamples` and `EconlibTest` are built with the same Lake project and are intended to compile with the main library as part of the default build.

Source code is available under the Apache 2.0 license (for `Mathlib` compatibility) on GitHub. Rendered API documentation and search tools are available through the project site: Type-shape search at `lean4.party/loogle`, semantic search at `lean4.party/search`, and generated documentation at `lean4.party/docs`.

Bibliography

[1] J. D. Angrist and J.-S. Pischke, "The Credibility Revolution in Empirical Economics: How Better Research Design is Taking the Con out of Econometrics," *Journal of Economic Perspectives*, vol. 24, no. 2, pp. 3–30, 2010, doi: 10.1257/jep.24.2.3.

[2] L. Vilhuber, "Reproducibility and Replicability in Economics," *Harvard Data Science Review*, vol. 2, no. 4, 2020, doi: 10.1162/99608f92.4f6b9e67.

[3] G. Christensen and E. Miguel, "Transparency, Reproducibility, and the Credibility of Economics Research," *Journal of Economic Literature*, vol. 56, no. 3, pp. 920–980, 2018, doi: 10.1257/jel.20171350.

[4] G. King, "Replication, Replication," *PS: Political Science & Politics*, vol. 28, no. 3, pp. 444–452, 1995, doi: 10.2307/420301.

[5] A. Lupia and C. Elman, "Openness in Political Science: Data Access and Research Transparency," *Political Science and Politics*, vol. 47, no. 1, pp. 19–42, 2014, doi: 10.1017/s1049096513001716.

[6] K. J. Arrow, *Social choice and individual values*. New York: Wiley, 1951, p. xi, 99 pages.

[7] J. H. Blau, "The Existence of Social Welfare Functions," *Econometrica*, vol. 25, no. 2, p. 302, 1957, doi: 10.2307/1910256.

[8] K. J. Arrow, *Social Choice and Individual Values*. Wiley, 1963.

[9] J. Geanakoplos, "Three brief proofs of Arrow?s Impossibility Theorem," *Economic Theory*, vol. 26, no. 1, pp. 211–215, 2005, doi: 10.1007/s00199-004-0556-7.

[10] T. Nipkow, "Social Choice Theory in HOL," *Journal of Automated Reasoning*, vol. 43, no. 3, pp. 289–304, 2009, doi: 10.1007/s10817-009-9147-4.

[11] J. W. Hatfield and P. R. Milgrom, "Matching with Contracts," *American Economic Review*, vol. 95, no. 4, pp. 913–935, 2005, doi: 10.1257/0002828054825466.

[12] O. Aygün and T. Sönmez, "Matching with Contracts: Comment," *American Economic Review*, vol. 103, no. 5, pp. 2050–2051, 2013, doi: 10.1257/aer.103.5.2050.

[13] Y. J. Levy and A. McLennan, "Corrigendum to "Discounted Stochastic Games With No Stationary Nash Equilibrium: Two Examples," *Econometrica*, vol. 83, no. 3, pp. 1237–1252, 2015, doi: 10.3982/ecta12183.

[14] A. Citanna, D. Gottlieb, P. Siconolfi, and X. Zhang, "Corrigendum: Long-Term Contracting With Time-Inconsistent Agents," *Econometrica*, vol. 91, no. 3, pp. 25–30, 2023, doi: 10.3982/ecta21301.

[15] P. J. Reny, "On the Existence of Pure and Mixed Strategy Nash Equilibria in Discontinuous Games," *Econometrica*, vol. 67, no. 5, pp. 1029–1056, 1999, doi: 10.1111/1468-0262.00069.

[16] T. `mathlib` Community, "The Lean mathematical library," *arXiv*, 2019, doi: 10.48550/ARXIV.1910.09336.

[17] L. de Moura and S. Ullrich, "The Lean 4 Theorem Prover and Programming Language," Springer International Publishing, 2021, pp. 625–635. doi: 10.1007/978-3-030-79876-5_37.

[18] D. Austen-Smith and J. S. Banks, *Positive Political Theory I*. University of Michigan Press, 2000.

[19] K. J. Arrow, "An Extension of the Basic Theorems of Classical Welfare Economics," University of California Press, 1951, pp. 507–532.

[20] G. Debreu, *Theory of Value: An Axiomatic Analysis of Economic Equilibrium*. Wiley, 1959.

[21] L. W. McKenzie, "On the Existence of General Equilibrium for a Competitive Market," *Econometrica*, vol. 27, no. 1, p. 54, 1959, doi: 10.2307/1907777.

[22] T. Tao, "Machine-Assisted Proof," *Notices of the American Mathematical Society*, vol. 72, no. 01, 2025, doi: 10.1090/noti3041.

[23] J. Commelin and A. Topaz, "Abstraction boundaries and spec driven development in pure mathematics," 2023, doi: 10.48550/ARXIV.2309.14870.

[24] D. Gottlieb and X. Zhang, "Long-Term Contracting With Time-Inconsistent Agents," *Econometrica*, vol. 89, no. 2, pp. 793–824, 2021, doi: 10.3982/ecta17126.

[25] X. Bei, J. Ma, Z. Jing, H. Fu, and Z. G. Tang, "EconCSLib: A Lean Library for Computational Economics and AI-Assisted Research," 2026, doi: 10.48550/ARXIV.2606.16144.

[26] C. Li, Z. Wang, W. He, Y. Wu, S. Xu, and Z. Wen, "Formalization of Complexity Analysis of the First-order Algorithms for Convex Optimization," 2024, doi: 10.48550/ARXIV.2403.11437.

[27] E. Kamenica and M. Gentzkow, "Bayesian Persuasion," *American Economic Review*, vol. 101, no. 6, pp. 2590–2615, 2011, doi: 10.1257/aer.101.6.2590.

[28] P. Dworczak and A. Kolotilin, "The persuasion duality," *Theoretical Economics*, vol. 19, no. 4, pp. 1701–1755, 2024, doi: 10.3982/te5900.

[29] harfe, "Kakutani and Brouwer Fixed-Point Theorems Formally Proven in Lean 4." [Online]. Available: https://github.com/harfe/fixed-point-theorems-lean4

[30] F. Wiedijk, "Formalizing Arrow's theorem," *Sadhana*, vol. 34, no. 1, pp. 193–220, 2009, doi: 10.1007/s12046-009-0005-1.

[31] U. Grandi and U. Endriss, "First-Order Logic Formalisation of Arrow's Theorem," Springer Berlin Heidelberg, 2009, pp. 133–146. doi: 10.1007/978-3-642-04893-7_11.

[32] W. H. Holliday, C. Norman, and E. Pacuit, "Voting Theory in the Lean Theorem Prover," 2021, doi: 10.48550/ARXIV.2110.08453.

[33] P. Tang and F. Lin, "Computer-aided proofs of Arrow's and other impossibility theorems," *Artificial Intelligence*, vol. 173, no. 11, pp. 1041–1053, 2009, doi: 10.1016/j.artint.2009.02.005.

[34] C. Geist and U. Endriss, "Automated Search for Impossibility Theorems in Social Choice Theory: Ranking Sets of Objects," *Journal of Artificial Intelligence Research*, vol. 40, pp. 143–174, 2011, doi: 10.1613/jair.3126.

[35] F. Brandl, F. Brandt, M. Eberl, and C. Geist, "Proving the Incompatibility of Efficiency and Strategyproofness via SMT Solving," *Journal of the ACM*, vol. 65, no. 2, pp. 1–28, 2018, doi: 10.1145/3125642.

[36] R. Vestergaard, "A constructive approach to sequential Nash equilibria," *Information Processing Letters*, vol. 97, no. 2, pp. 46–51, 2006, doi: 10.1016/j.ipl.2005.09.010.

[37] S. Le Roux, "Acyclic Preferences and Existence of Sequential Nash Equilibria: A Formal and Constructive Equivalence," Springer Berlin Heidelberg, 2009, pp. 293–309. doi: 10.1007/978-3-642-03359-9_21.

[38] J. Parsert and C. Kaliszyk, "Towards Formal Foundations for Game Theory," Springer International Publishing, 2018, pp. 495–503. doi: 10.1007/978-3-319-94821-8_29.

[39] math-xmum, "GameTheory: A Lean Formalization of Game Theory." [Online]. Available: https://github.com/math-xmum/gametheory

[40] C. Lange, C. Rowat, and M. Kerber, "The ForMaRE Project - Formal Mathematical Reasoning in Economics," 2013, doi: 10.48550/ARXIV.1303.4194.

[41] M. B. Caminati, M. Kerber, C. Lange, and C. Rowat, "Sound Auction Specification and Implementation," ACM, 2015, pp. 547–564. doi: 10.1145/2764468.2764511.

[42] G. Barthe, M. Gaboardi, E. J. G. Arias, J. Hsu, A. Roth, and P.-Y. Strub, "Computer-aided verification in mechanism design," *arXiv*, 2015, doi: 10.48550/ARXIV.1502.04052.

[43] C. Kaliszyk and J. Parsert, "Formal microeconomic foundations and the first welfare theorem," ACM, 2018, pp. 91–101. doi: 10.1145/3167100.

[44] P. Koepke and P. Schäfer, "Formalizing the Solow Model in \mathbb Naproche", Springer Nature Switzerland, 2026, pp. 357–370. doi: 10.1007/978-3-032-07021-0_20.

[45] M. Garg and S. Sarswat, "The Design and Regulation of Exchanges: A Formal Approach," 2022, doi: 10.48550/ARXIV.2210.05447.

[46] C. Barrett *et al.*, "CSLib: The Lean Computer Science Library," 2026, doi: 10.48550/ARXIV.2602.04846.

[47] M. Kerber, C. Lange, and C. Rowat, "An Introduction to Mechanized Reasoning," *arXiv*, 2016, doi: 10.48550/ARXIV.1603.02478.

[48] J. Avigad, "Mathematics and the formal turn," 2023, doi: 10.48550/ARXIV.2311.00007.

[49] P. Scholze, "Half a Year of the Liquid Tensor Experiment: Amazing Developments." 2021.

[50] N. Garg, "EconCSLib: AI-Assisted Lean Formalization for Economics & Computation research," 2026, doi: 10.48550/ARXIV.2606.13306.

[51] Y. Lin *et al.*, "Goedel-Prover: A Frontier Model for Open-Source Automated Theorem Proving," 2025, doi: 10.48550/ARXIV.2502.07640.

[52] S. Varambally, T. Voice, Y. Sun, Z. Chen, R. Yu, and K. Ye, "Hilbert: Recursively Building Formal Proofs with Informal Reasoning," 2025, doi: 10.48550/ARXIV.2509.22819.

A Headline Results by Module

This appendix indexes the headline declarations of each `Econlib` module: The primitive types that fix a module's interface and the marquee theorems a reader is likely to recognize by name. It is a guide to the library's surface, not a complete inventory; the full list of declarations, together with the worked examples and numerical witnesses, is available in the online docs.

A.1 Preferences

Declaration	Statement
`PreferenceRel`	The primitive: a complete, transitive weak preference relation on an arbitrary outcome type.

`preferenceOfRealUtility`	Turns any real-valued utility into a preference relation, with no added hypotheses.
`RepresentsRealPreference`	The predicate that a utility function represents a given preference relation.
`exists_utility_representation_of_finite`	Any preference relation on a finite outcome set admits a real-valued utility representation.
`exists_continuous_utility_representation`	Debreu's theorem: a continuous preference on a second-countable space has a continuous utility.
`SinglePeakedRel`	Single-peaked preferences: a peak alternative with utility falling monotonically to either side.
`risk_averse_iff_absoluteRiskAversion_nonneg`	Arrow–Pratt: risk aversion is equivalent to a nonnegative absolute risk-aversion coefficient.
`prudent_iff_iteratedDeriv3_nonneg`	Prudence is equivalent to a nonnegative third derivative of the utility function.
`ConstantRelativeRiskAversionUtility`	The CRRA functional form, bundled with its risk-attitude consequences.
`ConstantAbsoluteRiskAversionUtility`	The CARA (exponential) functional form, carrying constant absolute risk aversion.

A.2 Probability

Declaration	Statement
`FinDist`	Finite probability mass function whose expectation is a literal finite sum.
`CountDist`	Countably supported distribution over an encodable type.
`ContDist`	Absolutely continuous real distribution bundling a density and its CDF.
`MixedDist`	Finitely many atoms together with an absolutely continuous part on the reals.
`ProbDist`	General probability measure, the abstract carrier for stochastic orders (alias of `Mathlib`'s measure).
`FOSD_iff_expect_mono`	First-order stochastic dominance holds iff every monotone payoff has higher expected value.
`SOSD.iff_expect_concave`	Second-order dominance holds iff every monotone concave utility prefers the dominating law.
`isMPS_iff_shortfall`	A mean-preserving spread is characterized by dominance of expected shortfalls, that is, the convex order.
`strassen_iff`	Strassen: on compact support the convex order is equivalent to a martingale coupling.

`HasMLRP`	Monotone likelihood ratio property for a parameterized family, forcing dominance-ordered posteriors.
`CountDist.posterior`	Bayesian posterior from a prior and likelihood, re-entering the carrier it came from.
`geometric_convergence`	A positive finite Markov chain converges geometrically to its unique stationary distribution.
`gaussian_posterior`	Normal–normal signal extraction: a Gaussian prior and signal yield a Gaussian posterior.
`betaWithMean_convexOrder`	At a fixed mean, less concentrated Beta laws dominate more concentrated ones in convex order.

A.3 Optimization

Declaration	Statement
`argmaxRel`	Ordinal argmax: the greatest elements of a preference relation over a feasible set.
`argmax`	Cardinal argmax: the maximizers of a real objective over a constraint set.
`valueFunction`	The value function, the supremum of the objective over the feasible set.
`argmax_eq_argmaxRel_of_represents`	Cardinal and ordinal argmax agree when the utility represents the preference.
`valueFunction_continuous`	Berge (i): the value function is continuous under a continuous objective and correspondence.
`argmax_upperHemicontinuous`	Berge (ii): the argmax correspondence is upper hemicontinuous with compact, continuous data.
`argmax_strongSetOrder_of_weakSingleCrossing`	Milgrom–Shannon: argmax sets rise in the strong set order under single crossing.
`MaxKKT.isMaxOn`	KKT sufficiency: a certified first-order point is a global constrained maximum.
`hasGradientAt_profitFunction`	Hotelling's lemma: the price-gradient of the profit function is profit-maximizing supply.
`hasGradientAt_negExpendFunction`	Shephard's lemma: the price-gradient of the expenditure function is compensated demand.
`bellmanOperator_contraction`	Blackwell: the Bellman operator is a sup-norm contraction at the discount modulus.
`value_iteration_converges`	Value iteration from any bounded guess converges in sup norm to the value function.

`WeightedBlackwell`	Weighted-norm dynamic-programming layer admitting unbounded logarithmic and CRRA rewards.
`krDist_eq_krTransportCost`	Kantorovich–Rubinstein duality for the Wasserstein-1 optimal-transport problem.

A.4 Game Theory

Declaration	Statement
`EquilibriumProblem`	The shared spine: strategies, deviators, a legal-deviation relation, and a value function.
`IsEquilibrium`	No deviator can move to a legal alternative with strictly higher value.
`GameMorphism.transport`	Carries equilibria across a value- and deviation-preserving map between games.
`NashExistenceData.exists_equilibrium`	Kakutani existence from compact convex strategy slices and continuous best replies.
`exists_mixedNash`	Nash's theorem: every finite strategic game has a mixed-strategy equilibrium.
`exists_correlatedEq`	Every finite strategic game admits a correlated equilibrium.
`exists_isDistBNE`	Milgrom–Weber: existence of a distributional Bayes–Nash equilibrium in continuous-type games.
`BehavioralStrategy`	A strategy indexed by information sets, so information constraints cannot be violated.
`IsPerfectRecall`	Players never forget their own past observations or actions.
`mixed_realizes_behavioral`	Kuhn's theorem: under perfect recall mixed and behavioral strategies are realization-equivalent.
`IsSequentialEquilibrium`	A sequentially rational assessment with consistent limit-of-full-support beliefs.
`IsSequentialEquilibrium_of_oneShot`	One-shot deviation: no profitable one-step deviation gives a sequential equilibrium.
`exists_signalingSequentialEquilibrium`	Under a full-support prior, every finite signaling game has a sequential equilibrium.
`SurvivesIntuitiveCriterion`	The Cho–Kreps intuitive criterion, pruning equilibria on equilibrium-dominated beliefs.
`shapleyValue_unique`	Efficiency, symmetry, dummy, and additivity uniquely determine the Shapley value.

`core_nonempty_iff_balanced`	Bondareva–Shapley: a transferable-utility game has a nonempty core iff it is balanced.

A.5 Mechanism Design

Declaration	Statement
`IndirectMechanism.IsBNE_iff`	A mechanism's Bayes–Nash equilibria are exactly the interim no-deviation profiles.
`directify_isBIC`	Revelation principle: every equilibrium induces an equivalent incentive-compatible direct mechanism.
`vcgMechanism_isDSIC`	The VCG (Clarke-pivot) mechanism is dominant-strategy incentive compatible.
`myerson_lemma`	Myerson's lemma: a single-parameter rule is implementable iff its allocation is monotone.
`revenue_equivalence`	Mechanisms sharing an allocation and lowest-type utility charge identical expected payments.
`expected_virtualSurplus_le_ironed`	Ironing: ironed virtual surplus dominates raw virtual surplus in expectation.
`exists_optimal_auction_regular`	Under regularity a feasible auction attains the virtual-surplus revenue bound.
`firstPriceBid_isBNE`	The symmetric first-price bid schedule is a Bayes–Nash equilibrium.
`myerson_satterthwaite`	Myerson–Satterthwaite: no bilateral mechanism is efficient, incentive compatible, individually rational, and budget balanced.
`SignalStructure`	A finite experiment mapping states to distributions over messages in Bayesian persuasion.
`concaveClosure`	The concave closure of the value function, giving the sender's optimal persuasion value.
`caratheodory_simplex`	A finite persuasion problem needs at most one more signal than there are states.
`Duality.weakDuality`	The persuasion value is bounded above by the dual price-function value.
`strongDuality_of_isKRLipschitz`	KR strong duality: for KR-Lipschitz objectives the primal and dual values coincide and are attained.

A.6 Social Choice

Declaration	Statement

`Profile`	Assigns each voter a preference relation over the alternatives.
`WelfareFunction`	Maps profiles in its domain to a social ranking.
`ChoiceFunction`	Maps profiles to a nonempty set of winning alternatives.
`arrow_impossibility`	Arrow: on the universal domain, Weak Pareto and IIA force a dictator.
`arrow_impossibility_strict_domain`	Arrow on the strict-preference domain, the form Gibbard–Satterthwaite consumes.
`gibbard_satterthwaite`	Every strategy-proof, surjective resolute rule on three or more alternatives is dictatorial.
`may_characterization`	May: anonymity, neutrality, and positive responsiveness characterize majority rule on two alternatives.
`exists_condorcetWinner_of_singlePeaked`	Black: an odd single-peaked electorate always has a Condorcet winner.
`WelfareFunction.WeakPareto`	Unanimous strict preference for one alternative over another makes society strictly prefer it.
`WelfareFunction.IIA`	Society's ranking of a pair depends only on voters' rankings of that pair.
`WelfareFunction.IsDictator`	A voter whose strict preferences society always adopts.
`ChoiceFunction.StrategyProof`	No voter gains by a unilateral misreport (with pessimistic and Kelly set-ranking variants).

A.7 Equilibrium

Declaration	**Statement**
`Economy`	Finite-agent exchange economy: preferences and nonnegative endowments over `L` goods.
`RegularEconomy`	Continuous, convex, monotone, desirable preferences with nonzero endowments.
`WalrasianEquilibrium`	Nonnegative nonzero prices, budget-optimal allocations, and free-goods market clearing.
`WalrasianEquilibriumWithTransfers`	Competitive equilibrium relative to balanced lump-sum wealth transfers.
`Irreducible`	McKenzie irreducibility: no coalition can be enriched without the complement contributing.
`walras_law`	The value of aggregate excess demand is zero at any prices.
`WalrasianEquilibrium.price_pos`	Every equilibrium price is strictly positive under local nonsatiation.

`exists_equilibrium`	Arrow–Debreu existence for a regular irreducible economy, via Kakutani and the irreducibility upgrade.
`WalrasianEquilibrium.paretoOptimal`	First Welfare Theorem: every equilibrium allocation is Pareto optimal.
`exists_walrasianEquilibriumWithTransfers`	Second Welfare Theorem: a Pareto optimum decentralizes after lump-sum transfers.
`ProductionEconomy`	Exchange economy extended with firms, technologies, and ownership shares.
`exists_equilibrium_prod`	Existence of Walrasian equilibrium with production under production irreducibility.
`demand_eq_singleton_of_cobbDouglas`	Closed-form Cobb–Douglas demand: each good's spending share equals its exponent share.
`roy_identity`	Roy's identity: Marshallian demand is the price-to-income gradient ratio of indirect utility.

A.8 Supporting Mathematics

Declaration	Statement
`brouwerFixedPoint`	Every continuous self-map of a nonempty compact convex set in finite dimensions has a fixed point.
`kakutaniFixedPoint`	Every closed-graph, convex-valued self-correspondence of a compact convex set has a fixed point.
`fanGlicksbergFixedPoint`	Kakutani extended to compact convex sets in locally convex spaces.
`cubicalSperner`	Any Sperner-proper coloring of the cubical grid has an odd number of fully colored cells.
`isCompact_convexHull_of_isCompact`	The convex hull of a compact set in finite dimensions is compact.
`subset_convexHull_extremePoints_of_compact_convex`	Minkowski: a compact convex set is the convex hull of its extreme points.
`AnalyticSet.exists_uniformization`	Jankov–von Neumann: an analytic relation with nonempty sections has a measurable selector.
`stieltjes_ibp`	Integration by parts for two monotone functions against their Stieltjes measures.

`Measurable.exists_isCompact_continuousOn`	Lusin: a measurable map is continuous off a set of arbitrarily small measure.