

A LEAN FORMALIZATION OF FINITE ZERO SUM MINIMAX VIA SION'S THEOREM

ARTHUR FREITAS RAMOS, DAVID BARROS HULAK, AND RUY J. G. B. DE QUEIROZ

ABSTRACT. We describe a Lean 4 formalization of the minimax theorem for finite nonempty two-player zero-sum games with arbitrary real payoffs. Mixed strategies are probability vectors, and the two game values are defined using real indexed suprema and infima. An explicit payoff bound supplies the boundedness obligations required by these conditional extrema. The reverse minimax inequality follows by applying Mathlib's proved Sion saddle-point theorem to compact convex coordinate simplices, with the minimizing player placed in its first argument. We explain the representation bridge, the proof obligations, and the passage from a saddle point to equality. The accompanying version-pinned artifact separates a reusable library from a statement-only challenge and its proved solution. This is an exposition of a finite-game specialization of existing formal analysis, with no claim of a new mathematical proof or formalization priority.

1. SCOPE AND MATHEMATICAL CONTEXT

Von Neumann's finite minimax theorem concerns two players with opposing interests who may randomize over their available actions [7]. The maximizing player's best guaranteed payoff equals the minimizing player's best guaranteed ceiling. The theorem also provides optimal mixed strategies. Here we explain a Lean formalization of that statement, including the details that connect a finite payoff matrix to the analytic infrastructure used in its proof.

The mathematical result is classical. Our proof route specializes the Sion saddle-point theorem already available in Mathlib [5, 1], whose source credits Antoine Chambert-Loir and Anatole Dedecker and describes its proof as following Komiyama [2]. We do not reconstruct von Neumann's original argument. Mathlib's general theorem carries the principal analytic existence result; the development under discussion supplies the finite-game representation, boundedness lemmas, simplex facts, application of that theorem, and a public minimax interface.

All implementation claims in this note refer to commit `f1d1d00591548c174407cd59ddc623a461c082c1` of the artifact [4]. Lean is pinned to `v4.35.0-rc2`. The Mathlib dependency is pinned to the same release tag and resolved to commit `065356127b1dc0016f66b7283ce0ce2c4055aa55`. The statements quantify over arbitrary finite nonempty pure-strategy types A and B and an arbitrary function $u : A \times B \rightarrow \mathbb{R}$. No rationality, sign, or extra regularity assumption is imposed on the payoffs. The first player maximizes u and the second minimizes it, so their payoffs may be interpreted as u and $-u$. The implementation stores just this one payoff function.

The exposition makes the formal theorem's meaning and dependencies inspectable: where nonemptiness is used, why real extrema require explicit bounds, and why the players change order at the Sion interface. Source alignment and ordinary scholarly review remain necessary for the surrounding prose and historical claims.

Date: September 30, 2026.

2020 Mathematics Subject Classification. 91A05, 68V20.

Key words and phrases. Finite zero-sum games, minimax theorem, Lean 4, Mathlib, Sion's theorem.

Author ORCID identifiers: Arthur Freitas Ramos, 0009-0003-3568-0325; David Barros Hulak, 0009-0002-8056-1774; Ruy J. G. B. de Queiroz, 0000-0003-1482-0977.

Copyright 2026 the authors. Licensed under Creative Commons Attribution 4.0 International (CC BY 4.0).

2. FINITE GAMES AND THEIR REAL VALUES

2.1. Probability vectors. For a finite nonempty set S , put

$$\Delta(S) = \left\{ z : S \rightarrow \mathbb{R} \mid (\forall s, 0 \leq z_s) \text{ and } \sum_{s \in S} z_s = 1 \right\}.$$

The public structure `Minimax.MixedStrategy` carries these data directly. In the following excerpts, mathematical symbols are rendered typographically from the Lean source.

```
public structure MixedStrategy (A : Type*)
  [Fintype A] [Nonempty A] where
  val : A → ℝ
  nonneg : ∀ a, 0 ≤ val a
  sum_one : ∑ a, val a = 1
```

The `Fintype` instances provide finite enumerations, while `Nonempty` excludes an empty action type. A uniform strategy has every coordinate equal to $|S|^{-1}$; since $|S| \neq 0$, its mass is one. This construction gives the instance `MixedStrategy.instNonempty` without selecting a particular pure action. The lemma `MixedStrategy.val_le_one` proves that each probability coordinate is at most one, by comparing it with the sum of all nonnegative coordinates.

The library also defines pure strategies and convex mixing. For $p, q \geq 0$ with $p + q = 1$, the mixture of $x, z \in \Delta(S)$ has coordinates $px_s + qz_s$. Both its nonnegativity and its mass-one condition are proved in the structure constructor. Mixed strategies themselves are not treated as a vector space: arbitrary addition or scalar multiplication need not preserve probability mass.

2.2. Payoff and bilinear extension. The expected payoff at independent mixed strategies is

$$(1) \quad P_u(x, y) = \sum_{a \in A} \sum_{b \in B} x_a y_b u(a, b), \quad x \in \Delta(A), \quad y \in \Delta(B).$$

This is `Minimax.expPayoff`. Its unrestricted extension `Minimax.weightedPayoff` uses the same formula on $(A \rightarrow \mathbb{R}) \times (B \rightarrow \mathbb{R})$. Additivity and scalar compatibility are proved in each variable by finite-sum identities. The extension is bilinear; its restriction to each simplex is affine. The identity `expPayoff_eq_weightedPayoff` is definitional, so passing from a probability vector to its underlying coordinates does not require a mathematical approximation or an encoding change.

Pure strategies are Dirac vectors. The simplification lemmas `expPayoff_pure_left`, `expPayoff_pure_right`, and `expPayoff_pure_pure` identify their expected payoffs with the corresponding row average, column average, and matrix entry. For a constant payoff c , `expPayoff_const` gives expected payoff c . These are useful checks on the probability convention.

2.3. Bounds before extrema. Define the explicit finite bound

$$(2) \quad C_u = \sum_{a \in A} \sum_{b \in B} |u(a, b)|.$$

The declaration `payoffBound` uses this expression. Every matrix entry satisfies $|u(a, b)| \leq C_u$, because it is one nonnegative summand of the double sum. The weights $x_a y_b$ are nonnegative and have total mass one. Monotonicity of finite sums therefore gives

$$(3) \quad -C_u \leq P_u(x, y) \leq C_u \quad (x \in \Delta(A), \quad y \in \Delta(B)).$$

In the code this conclusion is `expPayoff_bounds`, obtained through `expPayoff_mono`, `le_expPayoff`, and `expPayoff_le`. The bound is deliberately simple rather than tight; the proof only needs a uniform finite interval.

The two values are

$$(4) \quad \underline{v}(u) = \sup_{x \in \Delta(A)} \inf_{y \in \Delta(B)} P_u(x, y), \quad \bar{v}(u) = \inf_{y \in \Delta(B)} \sup_{x \in \Delta(A)} P_u(x, y).$$

Their Lean definitions use indexed real extrema:

```
noncomputable def maxminValue (u : A → B → ℝ) : ℝ :=
  sup x : MixedStrategy A, inf y : MixedStrategy B,
    expPayoff u x y

noncomputable def minmaxValue (u : A → B → ℝ) : ℝ :=
  inf y : MixedStrategy B, sup x : MixedStrategy A,
    expPayoff u x y
```

Here the displayed definitions abbreviate their common finite and nonempty type hypotheses. In Mathlib, $\mathbb{R}$ is conditionally complete, not a complete lattice with real elements representing $\pm\infty$. Indexed extrema are total definitions, but the usual order laws must be applied with their nonemptiness and boundedness conditions. The uniform strategy witnesses nonempty indexing types, and (3) supplies bounded ranges for fixed-player payoffs. The lemmas `iInf_payoff_bounds` and `iSup_payoff_bounds` then bound the inner extrema; their ranges are bounded in turn. Consequently `maxminValue_bounds` and `minmaxValue_bounds` place both values in $[-C_u, C_u]$. Thus every extremum used in (4) has the intended real order-theoretic meaning.

3. THE ELEMENTARY INEQUALITY

Lemma 3.1. *For the game described above, $\underline{v}(u) \leq \bar{v}(u)$.*

Proof. For every $x \in \Delta(A)$ and $y \in \Delta(B)$,

$$\inf_{y' \in \Delta(B)} P_u(x, y') \leq P_u(x, y) \leq \sup_{x' \in \Delta(A)} P_u(x', y).$$

Taking the supremum over x and then the infimum over y gives the desired inequality. The nonempty bounded ranges established above justify these order operations in $\mathbb{R}$. $\square$

The implementation in `EasyDirection.lean` is a direct transcription. It uses `ciSup_le` and `le_ciInf` for the two outer comparisons, then composes `ciInf_le` with `le_ciSup`. The latter two calls explicitly use `bddBelow_payoff_right` and `bddAbove_payoff_left`. This direction requires no saddle-point theorem.

4. PREPARING THE SION APPLICATION

4.1. An ambient coordinate space. Sion's theorem is expressed for subsets of real topological vector spaces. A probability-vector structure does not directly provide that interface. The declaration `coordSimplexS` therefore defines $\Delta(S)$ as a subset of the ambient function space $S \rightarrow \mathbb{R}$, with its standard pointwise vector operations and product topology.

The map from `MixedStrategyS` to this set takes the field `val`; membership is given by the fields `nonneg` and `sum_one`. Conversely, `MixedStrategy.ofCoordSimplex` builds the probability-vector structure from a coordinate function and its membership proof. The resulting coordinates are definitionally the original function. The code needs these two conversions, rather than a separate transported topology on the structure.

Proposition 4.1. *The coordinate simplex $\Delta(S)$ is nonempty, convex, and compact.*

Proof. Nonemptiness follows from the uniform strategy. A convex combination preserves nonnegativity, and finite-sum distributivity gives $\sum_s (px_s + qz_s) = p + q = 1$.

For closedness, each constraint $0 \leq x_s$ defines a closed set because coordinate evaluation is continuous. Their intersection is closed. The finite coordinate sum is continuous, so its level set at 1 is closed as well. Their intersection is $\Delta(S)$.

Each coordinate of a point in the simplex lies in $[0, 1]$. Hence $\Delta(S)$ is a closed subset of the compact box $\prod_{s \in S} [0, 1]$, and is compact. $\square$

The corresponding declarations are `coordSimplex_nonempty`, `convex_coordSimplex`, `isClosed_coordSimplex`, and `isCompact_coordSimplex`. The compactness proof uses `isCompact_univ_pi`, `isCompact_Icc`, and the closed-subset compactness principle. Finiteness also makes the mass constraint a finite continuous sum.

4.2. Continuity and convexity of payoff sections. For each fixed y , the function $x \mapsto P_u(x, y)$ on the ambient coordinate space is a finite sum of coordinate evaluations multiplied by constants. It is continuous. The same argument applies to $y \mapsto P_u(x, y)$ for fixed x . These facts are proved as `continuous_weightedPayoff_left` and `continuous_weightedPayoff_right`; only separate continuity is needed for the application.

The additivity and scalar-compatibility lemmas from the basic library are packaged into two private linear maps, `payoffLeftLinearMap` and `payoffRightLinearMap`. A linear real-valued function is both convex and concave on a convex set, since its convex-combination inequality is an equality. Mathlib then supplies quasiconvexity and quasiconcavity, the weaker conditions used by Sion. Continuity similarly supplies the appropriate lower and upper semicontinuity on the simplices. These properties follow from the finite formula for every u ; they are not additional assumptions in the final game theorem.

5. FROM A SION SADDLE POINT TO MINIMAX

5.1. Player order at the library boundary. The real-valued declaration `Sion.exists_isSaddlePointOn` used here assumes nonempty compact convex sets X, Y ; each section in its first variable is lower semicontinuous and quasiconvex, and each section in its second variable is upper semicontinuous and quasiconcave. Its conclusion gives $a \in X$ and $b \in Y$ with

$$f(a, y) \leq f(x, b) \quad (x \in X, y \in Y).$$

This is the precise saddle-point interface in the pinned Mathlib source [1]. Both sets are compact in this interface. Sion's classical equality theorem has more general variants, including one-compact-set hypotheses [5]; those variants are unnecessary for this finite-game application.

The minimizing player occupies the first variable. In `HardDirection.lean`, the instantiation is

$$(5) \quad X = \Delta(B), \quad Y = \Delta(A), \quad f(y, x) = P_u(x, y).$$

Mathlib returns a minimizing coordinate vector y first and a maximizing vector x second. Converting them to mixed-strategy structures gives the following exact property.

Theorem 5.1 (Formalized saddle-point property). *There exist $x_* \in \Delta(A)$ and $y_* \in \Delta(B)$ such that*

$$(6) \quad P_u(x', y_*) \leq P_u(x_*, y') \quad \text{for all } x' \in \Delta(A), y' \in \Delta(B).$$

The declaration `Minimax.exists_payoff_saddle` states this property as follows, under the same common hypotheses:

$$\begin{aligned} \exists x : \text{MixedStrategy } A, \exists y : \text{MixedStrategy } B, \\ \forall x' : \text{MixedStrategy } A, \forall y' : \text{MixedStrategy } B, \\ \text{expPayoff } u \, x' \, y \leq \text{expPayoff } u \, x \, y' \end{aligned}$$

There are no admitted proof steps in this declaration. Sion's theorem is a proved dependency, not a new axiom introduced by the finite-game development.

5.2. The common value. Setting $y' = y_*$ and $x' = x_*$ in (6) separately gives the familiar optimality inequalities

$$(7) \quad P_u(x', y_*) \leq P_u(x_*, y_*) \leq P_u(x_*, y') \quad (x' \in \Delta(A), y' \in \Delta(B)).$$

Thus y_* is a minimizing best response to x_* and x_* is a maximizing best response to y_* .

Theorem 5.2 (Finite minimax). *For finite nonempty A, B and every real payoff $u : A \times B \rightarrow \mathbb{R}$,*

$$\sup_{x \in \Delta(A)} \inf_{y \in \Delta(B)} P_u(x, y) = \inf_{y \in \Delta(B)} \sup_{x \in \Delta(A)} P_u(x, y).$$

Proof. Choose x_*, y_* from Theorem 5.1 and set $v = P_u(x_*, y_*)$. Equation (7) implies

$$\bar{v}(u) \leq \sup_{x' \in \Delta(A)} P_u(x', y_*) \leq v \leq \inf_{y' \in \Delta(B)} P_u(x_*, y') \leq \underline{v}(u).$$

All real supremum and infimum comparisons are justified by the bounds in Section 2. Combining $\bar{v}(u) \leq \underline{v}(u)$ with Lemma 3.1 yields equality by antisymmetry. $\square$

The proof of `minimax_le_maxmin` follows exactly this chain. The final declaration is a single antisymmetry step:

```
public theorem minimax_theorem (u : A → B → ℝ) :
  maxminValue u = minmaxValue u :=
  le_antisymm (maxmin_le_minmax u) (minmax_le_maxmin u)
```

Its full common binder is

```
{A B : Type*} [Fintype A] [Fintype B]
[Nonempty A] [Nonempty B] (u : A → B → ℝ)
```

In particular, A and B may be distinct finite types.

Remark 5.3 (Attainment and existence). Equation (7), together with the strategies themselves as witnesses, also shows that the displayed inner extrema at x_*, y_* equal v , and that $v = \underline{v}(u) = \bar{v}(u)$. This is a mathematical consequence of the formal saddle-point property and the value definitions. The pinned public interface exports that property and minimax equality, rather than separate named declarations for every attainment identity. The development is noncomputable and does not supply an executable algorithm for finding optimal strategies from a numerical matrix.

5.3. A small interpretive example. To illustrate the distinction between pure and mixed play, consider the matching-pennies matrix

$$\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}.$$

If the maximizing player chooses the first row with probability p and the minimizing player chooses the first column with probability q , then the expected payoff is $(2p - 1)(2q - 1)$. At $p = q = 1/2$ both players guarantee value 0. Restricting to pure strategies instead gives `maxmin` -1 and `minmax` 1 , so the equality genuinely depends on allowing randomization. This calculation is an explanatory example here; the pinned artifact does not contain a dedicated formalized matching-pennies example.

6. ARTIFACT STRUCTURE AND VERIFICATION

6.1. Library and statement interface. The reusable proof is organized into three source modules. `Minimax/Basic.lean` defines strategies, payoff, mixing, pure embeddings, bounds, and values. `Minimax/EasyDirection.lean` proves the elementary inequality.

`Minimax/HardDirection.lean` establishes the coordinate-simplex interface and proves saddle existence, the reverse inequality, and equality. The top-level `Minimax.lean` imports the library.

The additional `Challenge.lean`, `Solution.lean`, and `comparator.json` files provide a Palomar-compatible statement/solution package. The challenge contains a copy of the strategy structure, three definition holes, and three theorem placeholders. Its six uses of `sorry` are intentional statement-surface placeholders. It imports only the relevant Mathlib modules and does not import the solution.

The solution imports the actual library and restates the three proved results in namespace `Minimax.Palomar`, using the corresponding `Minimax` library theorems. `Challenge` and `Solution` are compiled in separate environments because their fully qualified statement names coincide. The comparator lists `expPayoff`, `maxminValue`, and `minmaxValue` as the three definitions, and the two inequalities plus equality as the three theorem targets. The strategy structure is copied in the challenge and is not listed as a definition target.

6.2. What is checked. The pinned script `scripts/verify-palomar.sh` builds the library and the challenge/solution modules, checks the comparator declarations and their kinds in the separate environments, and checks proof tokens and axiom dependencies. Its intended proof-bearing scope is the library and `Solution`, which contain no `sorry`; it treats the challenge’s six deliberate placeholders separately.

The source audit `M4_AUDIT.md` records the checked declaration signatures and reports that the library and the three solution theorems depend only on `propext`, `Classical.choice`, and `Quot.sound`. These are the permitted axioms recorded by the comparator. The verification report for the pinned submission, checked on September 30, 2026, reports a passing build and comparator and acceptance by Lean’s default kernel and the additional `con-ron` and `nanoda` checkers [3]. This dated result is evidence about that artifact version, rather than a promise about arbitrary future dependency versions or environments.

Kernel checking establishes the formal conclusions from the actual definitions and accepted foundations. It does not by itself certify the interpretation of an informal theorem, the accuracy of citations, authorship, novelty, or the contents of this exposition. Those require the source alignment described here and ordinary scholarly review. Likewise, a passing mechanical report does not assert repository registration or publication status.

6.3. Reproduction and provenance. For reproduction, retrieve the repository, check out the commit identified in Section 1, and use the pinned Lean and Mathlib environment. The artifact’s documented entry point is `scripts/verify-palomar.sh`; it includes setup specific to its managed execution environment. The accompanying README describes that setup, the separate challenge build, and the comparator invocation. The source code is licensed `BSD-3-Clause`; this statement describes the code license and does not prescribe the license selected for depositing this manuscript.

The artifact metadata records GPT-6.1 via Codex assistance for formalization and packaging. AI assistance was also used in preparing this exposition. The source metadata labels review as self-assessed. We do not infer a level of human understanding or independent human review from automated checks. No person-specific contribution statement is asserted in this note.

7. LIMITATIONS AND CONCLUSION

The formal result covers finite nonempty pure-strategy types, independent probability-vector mixed strategies, and a single arbitrary real payoff with opposing player objectives. It does not cover infinite pure-strategy sets, general multiplayer or non-zero-sum Nash equilibrium existence, correlated strategies, or an executable solver. Although the proof ingredient is a general analytic minimax theorem, the exported game statement has precisely the finite scope given above.

The specialization shows how a concise classical existence argument can be connected to a reusable formal interface. Probability structures provide the game model, explicit bounds make real extrema safe to manipulate, coordinate simplices provide the topological vector-space setting, and the Sion saddle point yields the reverse inequality. The resulting equality is therefore supported by a transparent chain from the finite payoff formula to the version-pinned library theorem.

ACKNOWLEDGMENTS

We acknowledge the Mathlib community [6] and, in particular, Antoine Chambert-Loir and Anatole Dedecker for the Sion implementation on which this development relies.

REFERENCES

1. Antoine Chambert-Loir and Anatole Dedecker, *Formalization of Sion’s version of the von Neumann minimax theorem*, Mathlib 4, `Mathlib/Topology/Sion.lean`, 2025, Pinned revision `065356127b1dc0016f66b7283ce0ce2c4055aa55`, accessed September 30, 2026. <https://github.com/leanprover-community/mathlib4/blob/065356127b1dc0016f66b7283ce0ce2c4055aa55/Mathlib/Topology/Sion.lean>.
2. Hidetoshi Komiya, *Elementary proof for Sion’s minimax theorem*, Kodai Mathematical Journal **11** (1988), no. 1, 5–7, <https://doi.org/10.2996/kmj/1138038812>.
3. PalomarRegistry, *Verification workflow for the minimax submission*, Workflow run 36719764602, 2026, Mechanical report checked September 30, 2026, for source revision `f1d1d00591548c174407cd59ddc623a461c082c1`. <https://github.com/PalomarRegistry/PalomarSubmission/actions/runs/36719764602>.
4. Arthur Freitas Ramos, *Von Neumann’s minimax theorem in Lean*, Version-pinned source repository, 2026, Revision `f1d1d00591548c174407cd59ddc623a461c082c1`, accessed September 30, 2026. <https://github.com/Arthur742Ramos/minimax-theorem-lean/tree/f1d1d00591548c174407cd59ddc623a461c082c1>.
5. Maurice Sion, *On general minimax theorems*, Pacific Journal of Mathematics **8** (1958), no. 1, 171–176, <https://doi.org/10.2140/pjm.1958.8.171>.
6. The mathlib Community, *The Lean mathematical library*, Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, ACM, 2020, <https://arxiv.org/abs/1910.09336>, pp. 367–381.
7. John von Neumann, *Zur Theorie der Gesellschaftsspiele*, Mathematische Annalen **100** (1928), 295–320, <https://doi.org/10.1007/BF01448847>.