Formalization of the Balinski–Young Theorem in Lean 4

MICHAŁ DOBRANOWSKI, AGH University, Poland

Apportionment is the problem of distributing a fixed number of seats among parties in proportion to their vote counts. Although it appears straightforward, the issue is filled with combinatorial paradoxes that have long troubled political systems. The Balinski–Young Theorem, a seminal result in apportionment theory, establishes a fundamental impossibility: no anonymous quota rule can avoid the population paradox. This paper presents a formalization of the Balinski–Young Theorem using the Lean 4 theorem prover. We introduce the necessary algebraic structures for apportionment rules, formalize the notions of quota compliance and population monotonicity, and present a formally verified proof of their incompatibility.

1 Introduction

The problem of distributing a fixed number of seats in parliament among different actors is common in political systems. In the United States, those actors are the states, which seek representation in the House of Representatives proportional to their populations. In many other countries, the actors are political parties, which seek representation in parliament proportional to the number of votes they received in the most recent election.

However, this problem is not as straightforward as it may seem, as evidenced by the fact that the United States has used five different rules for distributing seats throughout its history [3]. These rules, known as apportionment methods, differ in the way they translate population or vote counts into whole numbers of seats. Examples include the D'Hondt (Jefferson) method, the Sainte–Laguë (Webster) method, and the Hare–Niemeyer (Hamilton) method [1, 10].

In this work, we do not focus on these methods; instead, we examine a theorem by Balinski and Young [2], which shows that no apportionment method can simultaneously satisfy two very natural and desirable properties. We will define these properties and present a proof of the theorem in Sections 2 and 3.

The main contribution of this paper is the formalization of the Balinski–Young theorem in Lean 4 [9], a modern *interactive theorem prover* based on dependent type theory. Theorem provers allow for the precise encoding of mathematical definitions, statements, and proofs in a way that can be mechanically checked for correctness. Lean 4 is particularly notable because it is also a convenient programming language, which enabled the work described in Section 4 and opens the door to future implementations of mathematically verified apportionment methods, as discussed in Section 5.

Author's Contact Information: Michał Dobranowski, AGH University, Krakow, Poland, michal@dobranow.ski.

The full Lean 4 codebase, both the excerpts included in this paper and additional material omitted here for brevity, is available online as the *Apportionmentlib* library at https://github.com/mdbrnowski/apportionmentlib/tree/v0.6.0.

1.1 Related work

There is some previous work on the formalization of axiomatic social choice theory. The famous Arrow's impossibility theorem for single-winner voting was formalized by Wiedijk [14] in Mizar, by Nipkow [11] in Isabelle/HOL, and later by Souther and Davidson [13] in Lean 3. Many classical results, including Sen's paradox and May's theorem, were formalized by Gammie [6]. Additionally, Holliday et al. [8] provided an extensive theoretical framework for classical social choice theory. More recently, Eberl [5] formalized aspects of randomized social choice, including concepts such as stochastic dominance and random dictatorship.

To the best of the author's knowledge, the only work focusing on the formalization of committee voting is by Delemazure et al. [4], which—unlike the present work—addresses strategy-proofness rather than specific notions of proportionality.

2 Definitions

In this section, we define—both in natural language and in Lean—basic notions related to the apportionment process, such as elections, apportionment, apportionment rules, and properties of apportionment rules. All definitions follow those given in the textbook by Pukelsheim [12], although some notation may differ.

Definition 2.1. An election instance $E = (h, v)$ is a pair where h is the house size, i.e., the number of seats to distribute, and $v = (v_1, \ldots, v_n) \in \mathbb{N}^n$ is a *vote vector* representing the votes cast for each party. We require that the sum of votes be strictly positive.

```
structure Election (n : ℕ) where
  votes : Vector ℕ n
  houseSize : ℕ+
  votes_sum_pos : 0 < votes.sum
```

Note that we also require the house size to be positive. Pukelsheim [12] assumes only $h \in \mathbb{N}$, but requiring $h \in \mathbb{N}_+$ is necessary to properly define the notion of exactness. Although we do not define this notion here, it is nevertheless useful in other parts of apportionment theory.

The positivity of the number of parties n is omitted from the definition, as it follows from our existing assumptions. Indeed, we prove this as `Election.n_pos`.

Definition 2.2. An apportionment rule is a function $R : \mathbb{N}_+ \times \mathbb{N}^n \to \mathcal{P}(\mathbb{N}^n)$ that maps any election instance $E = (h, v)$ to a set of *seat vectors* representing the seats apportioned to each party, and satisfies the following properties:

(1) Non-emptiness: there exists at least one solution, i.e., $R(h, v) \neq \emptyset$;
(2) Inheritance of zeros: parties with zero votes are allocated zero seats, i.e., for every $x \in R(h, v)$ and $i \in \{1, \ldots, n\}$, if $v_i = 0$ then $x_i = 0$;
(3) House-size feasibility: the total number of allocated seats in each solution equals the house size, i.e., for every $x \in R(h, v)$ we have $\sum_{i=1}^{n} x_i = h$.

```
abbrev Apportionment (n : ℕ) : Type := Vector ℕ n

structure Rule where
  res : {n : ℕ} → Election n → Finset (Apportionment n)
  non_emptiness {n : ℕ} (election : Election n) : (res election).Nonempty
  inheritance_of_zeros {n : ℕ} (election : Election n) (i : Fin n) :
    election.votes[i] = 0 → ∀ App ∈ res election, App[i] = 0
  house_size_feasibility {n : ℕ} (election : Election n) :
    ∀ App ∈ res election, App.sum = election.houseSize
```

We define a rule to return a set of seat vectors rather than a single seat vector in order to handle ties. For example, when we want to apportion 3 seats and two parties receive the same number of votes, there is no deterministic, fair way to choose a unique allocation. In this case, the rule may return two possible solutions, e.g., $\{(1, 2), (2, 1)\}$. In large elections, ties are unlikely, so in most cases the set of seat vectors will be a singleton.

Two basic notions of apportionment fairness are anonymity and concordance. Anonymity requires that a rule treat all parties symmetrically, while concordance requires that a party with more votes should not receive fewer seats.

Definition 2.3. An apportionment rule is anonymous if permuting the votes of the parties permutes the allocation of seats in the same way.

```
class IsAnonymous (rule : Rule) : Prop where
  anonymous {n : ℕ} (election : Election n) (σ : Equiv.Perm (Fin n)) :
    let election' : Election n := election.mk_by_perm σ
    ∀ App, App ∈ rule.res election' ↔
      ∃ App' ∈ rule.res election, ∀ i, App[i] = App'[σ i]
```

Definition 2.4. An apportionment rule is concordant if whenever one party has fewer votes than another, it is allocated no more seats than that party.

```
class IsConcordant (rule : Rule) : Prop where
  concordant {n : ℕ} (election : Election n) (i j : Fin n) :
    election.votes[i] < election.votes[j] →
      ∀ App ∈ rule.res election, App[i] ≤ App[j]
```

Two additional notions of fairness are quota compliance and population monotonicity, which we define below. These properties are somewhat more restrictive, and as we will show, they are inherently incompatible.

Ideally, a party k should receive a fraction of all seats equal to the fraction of votes cast for that party, i.e., $h \cdot \frac{v_k}{\sum_i v_i}$ seats. This value is called the *quota* and is not necessarily an integer. It seems natural that parties should receive a number of seats that is close to their quota.

Definition 2.5. An apportionment rule is a quota rule if the number of seats allocated to each party is either the floor or the ceiling of its quota.

This definition is formalized in Lean as follows. We cast to $\mathbb{Q}$ to obtain a rational division result instead of performing integer division, which would truncate the value.

```
class IsQuotaRule (rule : Rule) : Prop where
  quota_rule {n : ℕ} (election : Election n) (i : Fin n) :
    let quota := (election.votes[i] * election.houseSize : ℚ) / election.votes.sum
    ∀ App ∈ rule.res election, App[i] = ⌊quota⌋ ∨ App[i] = ⌈quota⌉
```

Increasing the number of votes for a party can sometimes lead to counterintuitive outcomes. Consider two elections with the same house size h, but possibly different vote totals, represented by vote vectors v and v'. Formally, a *population paradox* occurs when the support rate for party i increases relative to that of party j, yet i loses seats while j gains seats. That is, when $v_i'/v_i > v_j'/v_j$, yet $x_i > x_i'$ and $x_j < x_j'$ for some $x \in R(h, v)$, $x' \in R(h, v')$.

Definition 2.6. An apportionment rule is *population monotone* if it does not admit the population paradox.

This definition is formalized as follows. To avoid casting to $\mathbb{Q}$, we multiply crosswise when comparing vote count ratios.

```
class IsPopulationMonotone (rule : Rule) : Prop where
  population_monotone {n : ℕ} (election₁ election₂ : Election n) (i j : Fin n) :
    election₁.houseSize = election₂.houseSize →
      election₂.votes[i] * election₁.votes[j] >
          election₂.votes[j] * election₁.votes[i] →
```

```
        ∀ App₁ ∈ rule.res election₁, ∀ App₂ ∈ rule.res election₂,
          ¬(App₁[i] > App₂[i] ∧ App₁[j] < App₂[j])
```

3 Proof of the Balinski–Young theorem

Now we prove the impossibility theorem. Our proof is a slightly simplified version of the original argument of Balinski and Young [2].

Lemma 3.1. *Every anonymous, population-monotone rule is concordant.*

We formalize the above statement as `IsConcordant_of_IsPopulationMonotone`:

```
lemma IsConcordant_of_IsPopulationMonotone (rule : Rule) [IsAnonymous rule]
    [IsPopulationMonotone rule] : IsConcordant rule := by ...
```

Proof. Let R be an anonymous, population-monotone rule, and let $E = (h, v)$ be an election instance. We show that $v_i < v_j \implies x_i \leq x_j$ for every $x \in R(E)$ and all $i, j \in \{1, \dots, n\}$.

Fix $x \in R(E)$ and let $i, j \in \{1, \dots, n\}$ be such that $v_i < v_j$. If no such pair i, j exists, the statement holds vacuously. Since R is anonymous, we may swap the votes of parties i and j, obtaining a new election in which the seat allocation swaps the corresponding components as well. Thus the resulting allocation x' satisfies $x'_i = x_j$ and $x'_j = x_i$.

During this swap, party i gains votes while party j loses votes. Since R is population-monotone, this change cannot cause party i to receive fewer seats. Hence $x_i \leq x'_i = x_j$. □

Theorem 3.2 (Balinski and Young [2]). *There is no anonymous quota rule that is population monotone.*

We formalize the above statement as `balinski_young`:

```
theorem balinski_young (rule : Rule) [IsAnonymous rule] [IsQuotaRule rule] :
    ¬IsPopulationMonotone rule := by ...
```

Proof. Let R be an anonymous quota rule. For the sake of contradiction, assume that R is also population monotone. Consider the election instance $E = (8, (660, 670, 2450, 6220))$. For ease of exposition, we visualize the situation in Table 1, denoting the quota of the i-th party by q_i.

Table 1. Election E with 8 seats to allocate.

	party 1	party 2	party 3	party 4
votes	660	670	2450	6220
q_i	0.53	0.54	1.96	4.98

Fix some $x \in R(E)$. Since R is a quota rule, $x_3 \leq 2$ and $x_4 \leq 5$. Since $h = 8$, this implies that $x_1 + x_2 \geq 1$. By Lemma 3.1 and quota compliance, it follows that $x_2 = 1$.

Now consider another election instance $E' = (8, (680, 675, 700, 6200))$, shown in Table 2, in which parties 1 and 2 gain support, while parties 3 and 4 lose support.

Table 2. Election E' with 8 seats to allocate.

	party 1	party 2	party 3	party 4
votes	680	675	700	6200
q_i	0.66	0.65	0.68	6.01

Again, fix some $x' \in R(E')$. Since R is a quota rule, $x'_4 \geq 6$, and thus $x'_1 + x'_2 + x'_3 \leq 2$. By applying Lemma 3.1 twice and using quota compliance, we obtain $x'_2 = 0$.

The fact that $x_2 > x'_2$ and $x_4 < x'_4$, while party 2 gains votes and party 4 loses votes, constitutes a population paradox, which contradicts the assumption of population monotonicity. □

4 Library development

Apart from formalizing several statements and their proofs, we have also developed useful tools to support future formalizations in apportionment theory. One such tool is *Plausible*, part of the de facto standard library for Lean 4, which allows users to quickly find counterexamples to the theorem currently being proved. For example, when given an obviously incorrect statement such as $\forall n \in \mathbb{N} : 2n < n + 5$, Plausible quickly finds a relatively small counterexample:

```
example : ∀ n : ℕ, 2*n < n + 5 := by
  plausible
  -- ====================
  -- Found a counter-example!
  -- n := 8
  -- issue: 16 < 13 does not hold
  -- (0 shrinks)
  -- --------------------
```

We extended the functionality of Plausible by adding support for our `Election` structure.

```
example (e : Election 4) : e.votes[0] ≤ 15 + e.votes[1] := by
  plausible
  -- ====================
  -- Found a counter-example!
  -- e := { votes := #[20, 0, 0, 0], houseSize := 1 }
```

```
  -- issue: 20 ≤ 15 does not hold
  -- (15 shrinks)
  -- -------------------

#sample Election 2
  -- { votes := #[1, 0], houseSize := 5 }
  -- { votes := #[1, 1], houseSize := 20 }
  -- { votes := #[3, 1], houseSize := 4 }
```

To achieve this, we defined instances of `Arbitrary` (allowing Lean to generate random elections) and `Shrinkable` (allowing Lean to minimize counterexamples). These instances needed to preserve two essential invariants: positivity of the house size and positivity of the total number of votes; both properties had to be proven, requiring additional reasoning about the implementation. The complete implementation is available in the `PlausibleInstances` module.

5 Future work

It is possible to drop the anonymity assumption from the Balinski–Young theorem. The proof by Gölz, Peters, and Procaccia [7] is longer than the original one, but it is not conceptually more difficult, so it should be possible to formalize this more general result as well. Furthermore, it would be valuable to implement apportionment rules, such as D'Hondt and Hamilton, in a provably correct way and to show that they constitute apportionment *methods*, i.e., that they satisfy five axioms: anonymity, balancedness, concordance, decency, and exactness (two of which have already been defined in this work). Such results would complement the theoretical development by connecting the axiomatic framework to concrete, verified procedures.

References

[1] Michel L. Balinski and Victoriano Ramírez. Parametric methods of apportionment, rounding and production. *Mathematical Social Sciences*, 37(2):107–122, 1999. doi: https://doi.org/10.1016/S0165-4896(98)00027-4.

[2] Michel L. Balinski and H. Peyton Young. *Fair Representation: Meeting the Ideal of One Man, One Vote.* Yale University Press, 1982. ISBN 0300027249.

[3] Charles M Biles. *The History of Congressional Apportionment.* Humboldt State University Press, 2021. ISBN 1947112031.

[4] Théo Delemazure, Tom Demeulemeester, Manuel Eberl, Jonas Israel, and Patrick Lederer. The incompatibility of strategy-proofness and representation in party-approval multi-winner elections. *Archive of Formal Proofs*, November 2022.

[5] Manuel Eberl. Randomised social choice theory. *Archive of Formal Proofs*, May 2016.

[6] Peter Gammie. Some classical results in social choice theory. *Archive of Formal Proofs*, November 2008.

[7] Paul Gölz, Dominik Peters, and Ariel D. Procaccia. In this apportionment lottery, the house always wins. *Operations Research*, 74(1):390–407, 2025. doi: 10.1287/opre.2022.0419.

[8] Wesley H. Holliday, Chase Norman, and Eric Pacuit. Voting theory in the Lean theorem prover. In *Logic, Rationality, and Interaction*, pages 111–127. Springer, 2021. ISBN 978-3-030-88707-0. doi: 10.1007/978-3-030-88708-7_9.

[9] Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, pages 625–635. Springer, 2021. doi: 10.1007/978-3-030-79876-5_37.

[10] Horst F. Niemeyer and Alice C. Niemeyer. Apportionment methods. *Mathematical Social Sciences*, 56(2):240–253, 2008. doi: https://doi.org/10.1016/j.mathsocsci.2008.03.003.

[11] Tobias Nipkow. Social choice theory in HOL: Arrow and Gibbard–Satterthwaite. *Journal of Automated Reasoning*, 43(3):289–304, 2009. doi: 10.1007/s10817-009-9147-4.

[12] Friedrich Pukelsheim. *Proportional Representation: Apportionment Methods and Their Applications*. Springer, 2017. doi: 10.1007/978-3-319-64707-4.

[13] Andrew Souther and Benjamin Davidson. Social choice theory in Lean. https://github.com/asouther4/lean-social-choice, 2021.

[14] Freek Wiedijk. Arrow's impossibility theorem. *Formalized Mathematics*, 15(4):171–174, 2007. doi: 10.2478/v10037-007-0020-9.