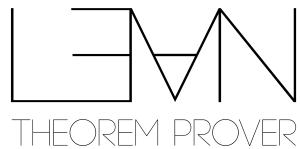
Automatic Geometry Theorem Proving

Using Polynomial Elaboration

Mauricio Barba da Costa
with Fabian Zaiser, Cameron Freer, Katie Collins, Josh Tenenbaum, and Vikash Mansinghka

October 2, 2025



Freek Wiedijk's 100 Theorems Challenge

Tracks progress of theorem provers in formalizing 100 classic theorems:

- Currently 82 theorems formalized in Lean

Source: `https://leanprover-community.github.io/100.html`

Missing Euclidean Geometry Theorems

Missing theorems from Freek Wiedijk's list of 100 theorems

These theorems are not yet formalized in Lean. Currently there are 18 of them. Among these, 0 have their statement formalized. Here is the list of the formalized theorems.

- 8: The Impossibility of Trisecting the Angle and Doubling the Cube
- 12: The Independence of the Parallel Postulate
- 13: Polyhedron Formula
- 21: Green's Theorem
- 28: Pascal's Hexagon Theorem
- 29: Feuerbach's Theorem
- 32: The Four Color Problem
- 33: Fermat's Last Theorem
- 41: Puiseux's Theorem
- 43: The Isoperimetric Theorem
- 47: The Central Limit Theorem
- 50: The Number of Platonic Solids
- 53: Pi is Transcendental
- 56: The Hermite-Lindemann Transcendence Theorem
- 61: Theorem of Ceva
- 84: Morley's Theorem
- 87: Desargues's Theorem
- 92: Pick's Theorem

Figure 1: Euclidean geometry theorems from Wiedijk's list not yet formalized in Lean

Related Work

Other work has also pointed out the deficiency of Mathlib's Euclidean geometry library and tried to expand it

LeanGeo and **LeanEuclid**

- Repositories formalizing *synthetic* Euclidean geometry via Jeremy Avigad's System E
- Axiomatic construction makes it difficult to port to Mathlib

Our approach complements these efforts with automated algebraic proving

Summary

Key contributions:

- Demonstrated algebraic automation of classical geometry theorems
- Identified an unsound theorem in an educational theorem-proving tool that we found in the process of formalizing a library of geometry theorems
- Reduced complex manual proofs to single tactic calls
- Outlined path toward `grind`-based automation in Mathlib
- Highlighted limitations and future research directions

From Geometry to Algebra

Key Insight: Euclidean geometry statements can often be proved using purely algebraic methods

Method:

1. Define geometric primitives using coordinates and polynomial constraints
2. Unfold definitions to obtain systems of polynomial equations
3. Apply polynomial solver (Gröbner basis)

This reduces geometric reasoning to algebraic computation

Existing Tools: JGEX and GeoGebra

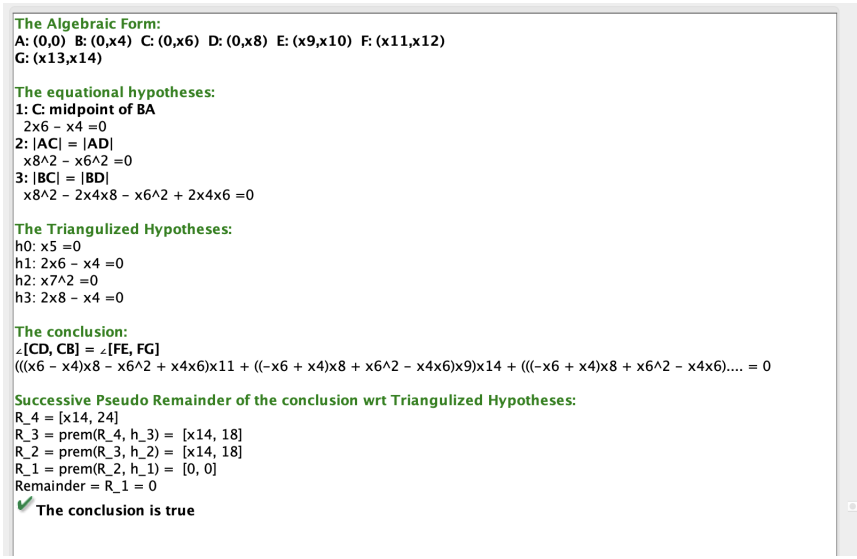
This algebraic approach is not new:

- **JGEX**: Automated geometry theorem prover
- **GeoGebra**: Educational geometry software

Problem: These tools are *not formally verified*

$\Rightarrow$ Can produce unexpected results!

Unsound Theorem in JGEX



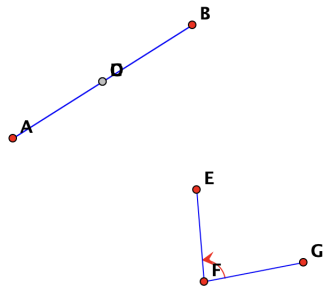
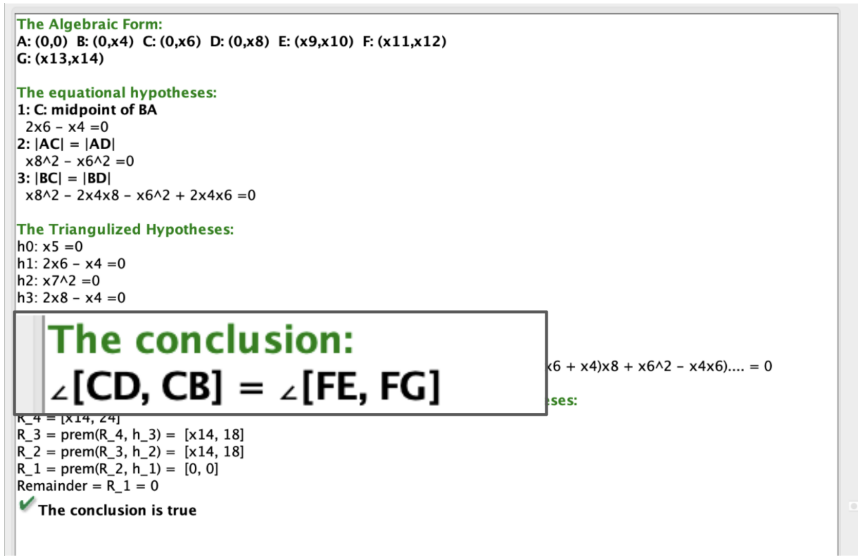


Figure 2: Construction of a degenerate angle equal to every other angle. The angle equality predicate is not transitive in this system

Unsound Theorem in JGEX



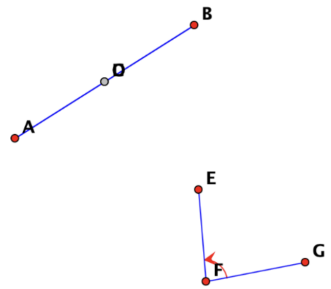


Figure 2: The conclusion of the statement is that two angles are equal to each other

Unsound Theorem in JGEX

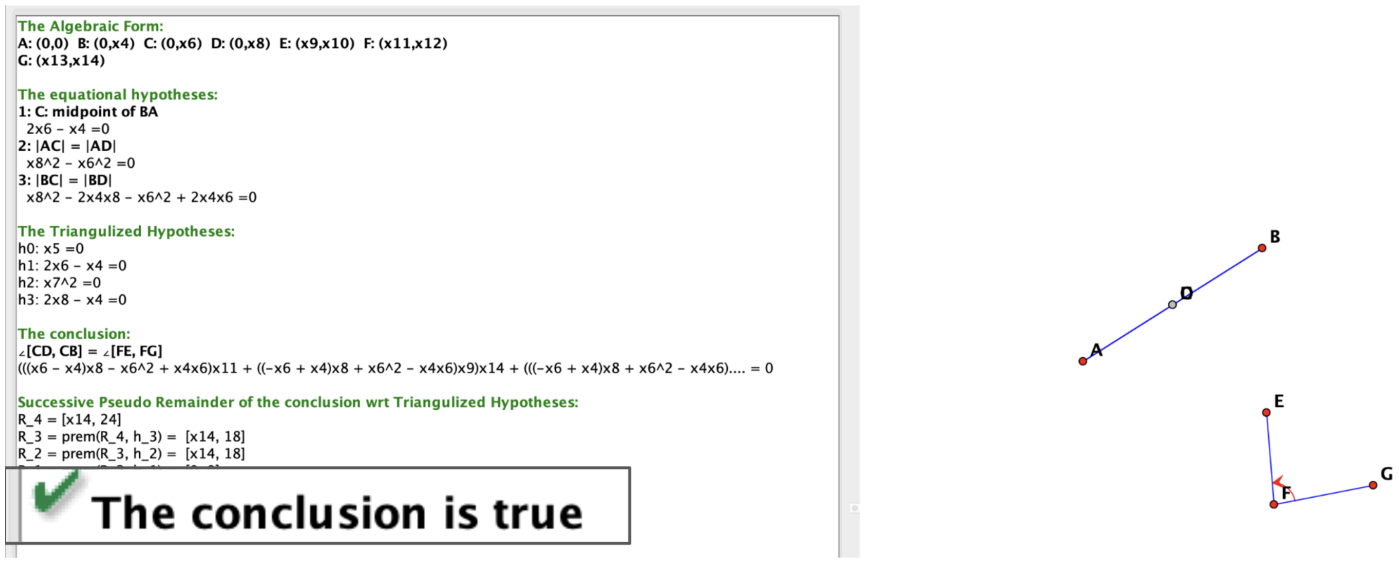


Figure 2: JGEX proves that the conclusion is true

Unsound Theorem in JGEX

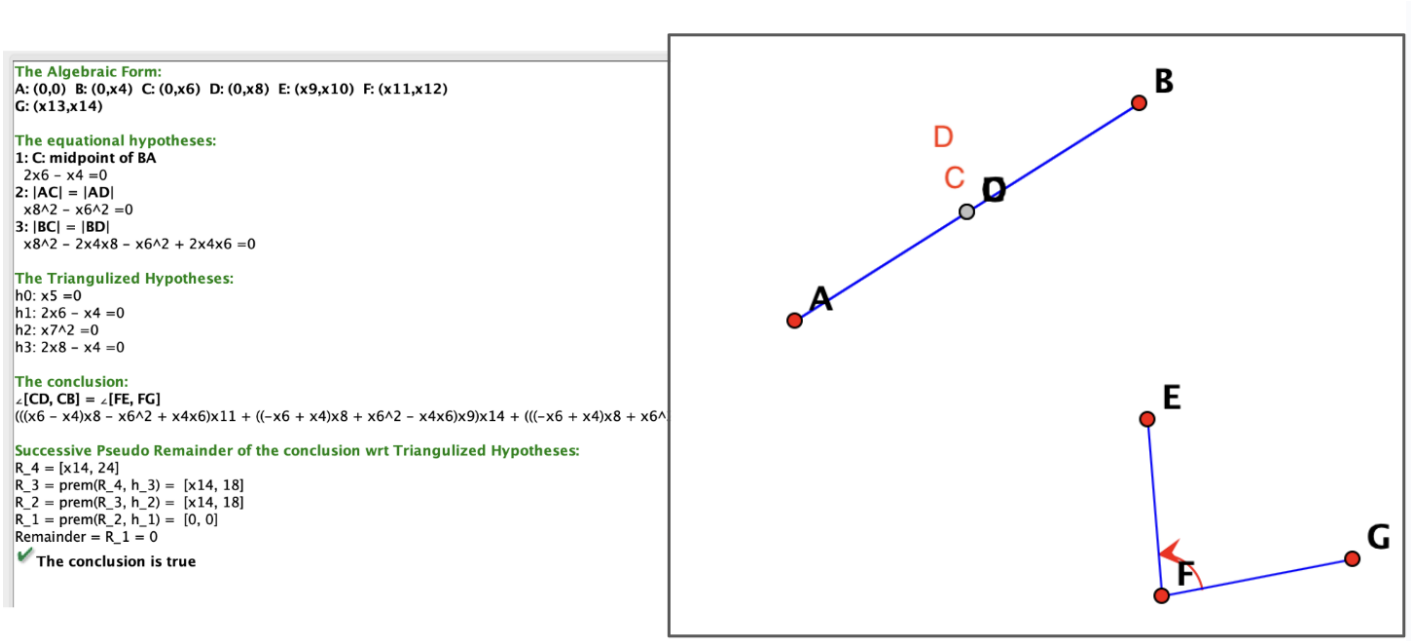


Figure 2: But, the angles D C B and E F G are clearly not the same

Examples of Converting Geometry to Algebra

```
@[euclid_simp]
def Collinear
  (p₁ p₂ p₃ : EuclidPoint)
  : Prop :=
  (p₂.x - p₁.x) *
  (p₃.y - p₁.y) -
  (p₂.y - p₁.y) *
  (p₃.x - p₁.x) = 0
```

```
@[euclid_simp]
def Rotate (P : EuclidPoint) (center :
     EuclidPoint) (u : EuclidPoint) :
     EuclidPoint :=
  let dx := P.x - center.x
  let dy := P.y - center.y
  EuclidPoint.mk
    (center.x + dx * u.x - dy * u.y)
    (center.y + dx * u.y + dy * u.x)
```

- These definitions can be simplified to polynomial equalities that depend only on the coordinates of points in $\mathbb{R}^2$
- The `@[euclid_simp]` tag tells Lean's elaborator how to recursively convert statements into polynomial expressions
- Lean's formal verification guarantees that this transformation is sound

The algebraic_euclid Tactic

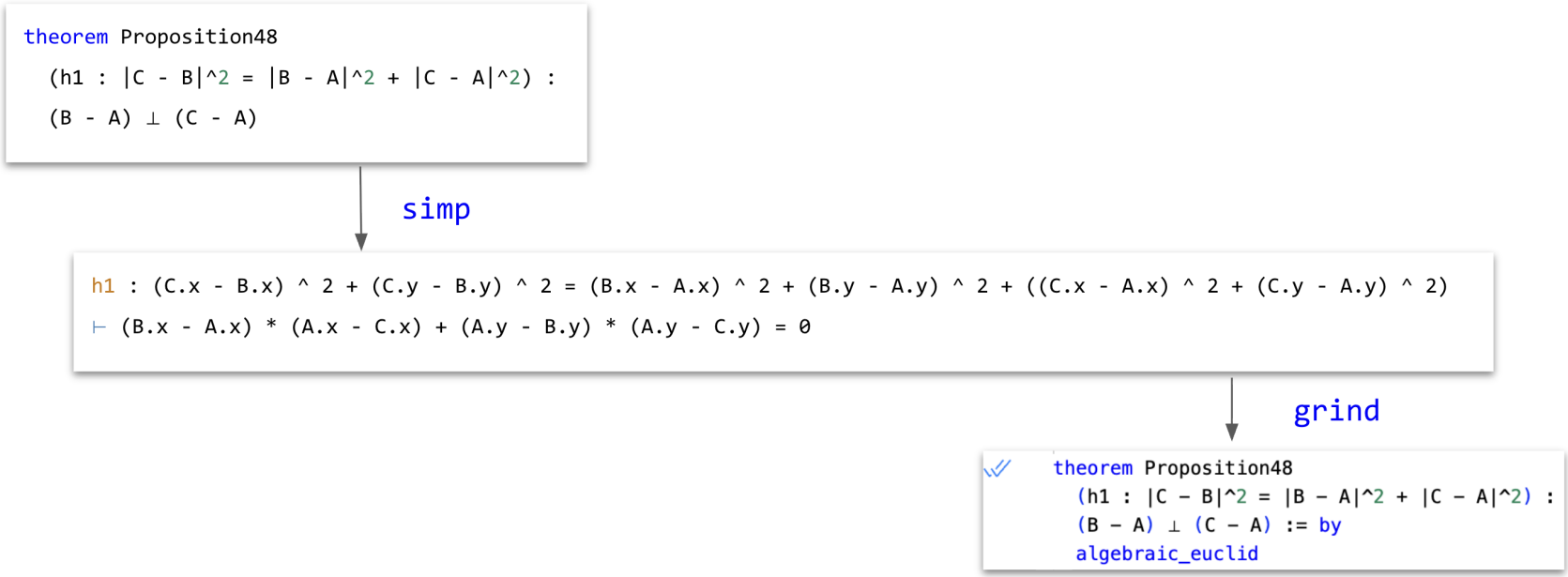


Figure 3: Converting the converse of the Pythagorean theorem into an algebraic statement and proving it using `grind`

Example: Pappus's Theorem

```
theorem Pappus_Theorem
  (h1 : Col A B C)
  (h2 : Col P Q R)
  (h4 : Col X A Q)
  (h5 : Col X B P)
  (h6 : Col Y A R)
  (h7 : Col Y C P)
  (h8 : Col Z B R)
  (h9 : Col Z C Q)
  (h10 : ¬(A - Q) || (B - P))
  :
  Col X Y Z
  := by algebraic_euclid
```

Example: Pappus's Theorem (Elaboration)

After elaboration, the geometric statement becomes:

```
h1 : (B.x - A.x) * (C.y - A.y) - (B.y - A.y) * (C.x - A.x) = 0
h2 : (Q.x - P.x) * (R.y - P.y) - (Q.y - P.y) * (R.x - P.x) = 0
h4 : (A.x - X.x) * (Q.y - X.y) - (A.y - X.y) * (Q.x - X.x) = 0
h5 : (B.x - X.x) * (P.y - X.y) - (B.y - X.y) * (P.x - X.x) = 0
h6 : (A.x - Y.x) * (R.y - Y.y) - (A.y - Y.y) * (R.x - Y.x) = 0
h7 : (C.x - Y.x) * (P.y - Y.y) - (C.y - Y.y) * (P.x - Y.x) = 0
h8 : (B.x - Z.x) * (R.y - Z.y) - (B.y - Z.y) * (R.x - Z.x) = 0
h9 : (C.x - Z.x) * (Q.y - Z.y) - (C.y - Z.y) * (Q.x - Z.x) = 0
h10 : ¬(Q.x - A.x) * (P.y - B.y) - (P.x - B.x) * (Q.y - A.y) = 0
⊢ (Y.x - X.x) * (Z.y - X.y) - (Y.y - X.y) * (Z.x - X.x) = 0
```

Pure polynomial algebra—ready for `grind`!

Almost 100 theorems that we've proven with a one-line proof in this way

Mathlib's Algebraic Foundation

- One reason the Euclidean geometry library in Mathlib might be so small is that existing Euclidean geometry theorems in Mathlib can be extremely cumbersome
- Good news: Mathlib's Euclidean geometry is already quite algebraic
- With careful setup, many Mathlib theorems become one-liners

Example: The sum of the angles of a triangle

Current Mathlib proof:

```
theorem angle_add_angle_add_angle_eq_pi
  {p₁ p₂ : P} (p₃ : P) (h : p₂ ≠ p₁) :
  ∠ p₁ p₂ p₃ + ∠ p₂ p₃ p₁ + ∠ p₃ p₁ p₂ = π := by
  rw [add_assoc, add_comm, add_comm (∠ p₂ p₃ p₁),
      angle_comm p₂ p₃ p₁]
  unfold angle
  rw [← angle_neg_neg (p₁ -ᵥ p₃),
      ← angle_neg_neg (p₁ -ᵥ p₂),
      neg_vsub_eq_vsub_rev, neg_vsub_eq_vsub_rev,
      neg_vsub_eq_vsub_rev, neg_vsub_eq_vsub_rev,
      ← vsub_sub_vsub_cancel_right p₃ p₂ p₁,
      ← vsub_sub_vsub_cancel_right p₂ p₃ p₁]
  exact angle_add_angle_sub_add_angle_sub_eq_pi _
    fun he => h (vsub_eq_zero_iff_eq.1 he)
```

Extensive manual rewriting required

Example: Stewart's Theorem

Current Mathlib proof:

```
/-- Stewart's Theorem. -/
theorem dist_sq_mul_dist_add_dist_sq_mul_dist
  (a b c p : P) (h : angle  b p c = π) :
  dist a b ^ 2 * dist c p + dist a c ^ 2 * dist b p =
  dist b c * (dist a p ^ 2 + dist b p * dist c p) := by
  rw [pow_two, pow_two, law_cos a p b, law_cos a p c,
      eq_sub_of_add_eq
        (angle_add_angle_eq_pi_of_angle_eq_pi a h),
      Real.cos_pi_sub,
      dist_eq_add_dist_of_angle_eq_pi h]
  ring
```

Multiple law of cosines applications and angle manipulations

Sum of Angles of a Triangle: Algebraic Version

With algebraic approach:

```
theorem Proposition32
  (h1 : |A - B| ≠ 0)
  (h2 : |A - C| ≠ 0)
  (h3 : |B - C| ≠ 0)
  (h4 : |B - A| ≠ 0)
  : ((∠ B A C (by nondegen)) + (∠ C B A (by nondegen)) +
  (∠ A C B (by nondegen))) = (⊥ + ⊥) := by
  algebraic_euclid
```

Single tactic call

Stewart's Theorem: Algebraic Version

With algebraic approach:

```
theorem StewartTheorem
  (h1 : Between B D C)
  : |A - D|^2 * |B - C| + |B - D| * |D - C| * |B - C| =
    |A - B|^2 * |D - C| + |A - C|^2 * |B - D|
  := by
    algebraic_euclid
```

Single tactic call

Why Current Mathlib Theorems Resist Automation

Issues preventing direct `grind` application:

1. Some definitions are adverse to polynomial solving. E.g. angles defined by radians cannot be not handled by purely polynomial solvers
2. Definitions also need explicit unfolding strategies

Solution: With targeted changes to definitions and `grind` configuration, many theorems become automatable

When the Method Fails

Fundamental limitations:

1. **Existential quantifiers**: Gröbner basis methods cannot handle statements requiring witness construction
2. **Ordering properties**: Statements relying on $<$, $\leq$ over $\mathbb{R}$ cannot be proved purely algebraically

   E.g. $x^2 \geq 0$ is necessary for many geometric theorems but can't be proved with a polynomial solver
3. **Computational complexity**: Gröbner basis is doubly exponential in number of variables
4. Difficult to diagnose `grind` failures:
   - Too many variables?
   - Missing ordering constraints?
   - Missing nondegeneracy condition?

Potential Workarounds

For ordering properties:

Tag relevant ordering theorems with `@[grind]` to help automation

For complexity:

Some problems simplified by fixing coordinate system (origin + unit point)

For existentials:

Provide explicit constructions, then use algebraic methods to verify properties

Future Work: Machine Learning Integration

Strongly scaffolded domain for LM-based theorem proving:

- Rapid conjecture generation and verification cycle
- Alternate between:
  - LM generates lemmas/subgoals
  - `grind` proves them algebraically
- Reinforcement learning to bootstrap geometry library

Euclidean geometry as a testbed for neural-symbolic theorem proving

Takeaways

1. A tactic similar to `algebraic_euclid` could be useful for proving Euclidean geometry statements in Mathlib
2. Modifying the Euclidean geometry library to work with `algebraic_euclid` requires thoughtful design choices
3. Many exciting directions for the `grind` tactic in Lean

References

Resources:

- Lean 100 Theorems: `leanprover-community.github.io/100.html`
- Wiedijk's List: `cs.ru.nl/~freek/100/`
- LeanGeo & LeanEuclid projects
- Mathlib documentation

Thank You!

Questions?

barba@mit.edu