Interactive Theorem Proving in Lean

Can We Teach Proofs to a Computer?

Alex Dolcos, Edward Kong, Lawrence Zhao, Nicholas Phillips
Vaibhav Karve (Team Leader)

University of Illinois at Urbana-Champaign

Illinois Geometry Lab
Summer Presentation
June 25 2020

Interactive Theorem Proving

An Interactive Theorem Prover is a program that provides a human user with software tools to assist with the development of formal proofs.

Curry-Howard equivalence

$$\textit{Mathematics} \cap \textit{Computer science}$$

Math: based on language of Propositions

CS: based on language of Types

Props are Types!

Use Type theory instead of Set theory

The Lean Theorem Prover

Lean Interactive Theorem Prover

- User defines proofs, computer checks their accuracy
- Computer correctly proves (disproves) statement

Curry-Howard continued . . .

Math	*Programming*
Objects	Terms
Sets	Types
Definitions	Functions
Lemmas/Theorems	Propositions (also a Type)
Proofs	Programs (also a Term)
Axioms	Constants

Motivations

We wanted a topic in mathematics that ...

- we are very familiar with ...
- Is not already in MATHLIB (Math library in Lean)
- For reference, Lean already understands set theory, group theory, number theory, and category theory.

Project Goals

1. Formalize axiomatic geometry of three types into the Lean programming language
   - Euclid's axioms ($\sim$ 300 BCE)
   - Hilbert's axioms ($\sim$ 1899)
   - Tarski's axioms ($\sim$ 1959)

2. Learn Lean syntax along the way to formalizing Geometry.

Figure: *source: Wikimedia Commons*

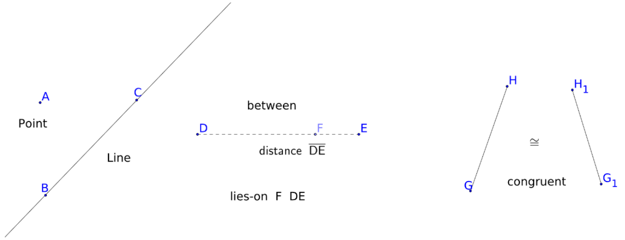
Euclidean Geometry

First proper foundation of geometry

Based on physical constructions with a compass and straightedge (points, lines and circles)

Defines a set of primitive objects – Does not utilize a coordinate system like Analytical Geometry

Starts with plane geometry and goes on to define 3-D solids

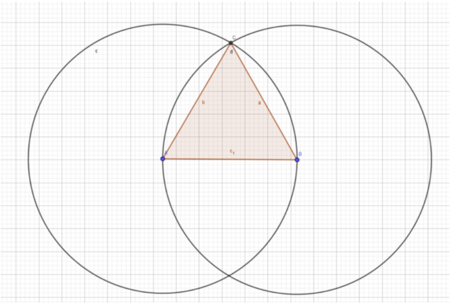


Formalizing Euclid in Lean

Formalizing Euclidean Geometry in Lean was challenging

- Missing axioms
- Verbose proofs

```
-- # Proposition 1
------------------
lemma construct_equilateral (s : Segment) : ∃ (tri: Triangle),
  s.p1 = tri.p1 ∧ s.p2 = tri.p2 ∧ is_equilateral tri :=
begin
  set c₁ : Circle := ⟨s.p1, s.p2⟩,
  set c₂ : Circle := ⟨s.p2, s.p1⟩,
  have h₁ := (hypothesis1_about_circles_radius s),
  have h₂ := hypothesis2_about_circles_radius s,
  set p : Point := circles_intersect c₁ c₂ h₁ h₂,
  have hp₁ : p ∈ circumference c₁, from (circles_intersect' c₁ c₂ h₁ h₂).1,
  have hp₂ : p ∈ circumference c₂, from (circles_intersect' c₁ c₂ h₁ h₂).2,
  use ⟨s.p1, s.p2, p⟩,
  --- Cleaning up the context ---
    tidy;
    unfold circumference radius_segment at hp₁ hp₂;
    unfold sides_of_triangle;
    dsimp * at *,
  --- Cleaning done ---
    {calc s.p1 · s.p2 ≃ s.p2 · s.p1 : by symmetry
                  ... ≃ s.p2 · p    : by assumption},
    {calc s.p2 · p ≃ s.p2 · s.p1 : by {apply cong_symm, assumption}
               ... ≃ s.p1 · s.p2 : by apply segment_symm
               ... ≃ s.p1 · p    : by assumption
               ... ≃     p · s.p1 : by symmetry},
```

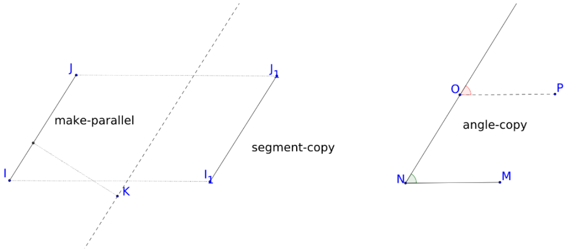


Hilbert's Geometry

A modern approach to Euclidean geometry

Set of 20 axioms (most of which relate to planar geometry)

Synthetic geometry, so it avoids using certain definitions in proofs (e.g. distance)

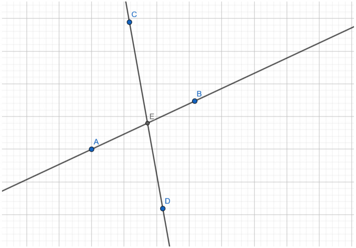


Formalizing Hilbert in Lean

We had to define many implicit relations and structures

Verbose, simple proofs on paper may be more difficult to prove to the computer

```
-- If two distinct lines intersect, then they do so in
exactly one point.
lemma single_intersection (l₁ l₂ : Line) :
     l₁ ≠ l₂
  → intersect_line l₁ l₂
  → ∃! x : Point, lies_on_line x l₁ ∧ lies_on_line x l₂ :=
begin
  intros h₁ h₂,
  rw intersect_line at h₂,
  choose x h₂ using h₂,
  use x,
  tidy,
  symmetry,
  by_contradiction,
  have line_exists := line_exists x y a_2,
    tidy,
  have h₃ : l₁ = ⟨x, y, a_2⟩,
    { apply line_unique, assumption, assumption},
  have h₄ : l₂ = ⟨x, y, a_2⟩,
    { apply line_unique, assumption, assumption},
  cc,
end
```



Challenges

- We needed a collaborative-editing platforms for Lean
- CoCalc, being the first platform, worked well, but was extremely slow when multiple people worked on it at once
- Bugs in Lean that required frequent restarts
- Missing axioms and propositions – particularly the ones that needed “length or distance” to be defined
- Writing proofs in Lean is easy, but making new (and correct) definitions is hard.

Future Directions

- Streamline definitions of Euclid's and Hilbert's axioms
- Euclid proves 48 propositions. So far, we have translated 2 of these to Lean
- Prove the Pythagorean Theorem in Lean
- Add Tarski's axioms and Birkhoff's axioms to Lean
- Add solid geometry (3-D), hyperbolic geometry and spherical geometry
- Explore other proof checkers like Coq and HOL-Light

Conclusion

We published our code on GitHub:
github.com/vaibhavkarve/leanteach2020.

Project documentation published on Illinois-Wiki:
wiki.illinois.edu/wiki/display/lt2020.

We thank David Frankel (Uni High class of 1976) whose gift made this experience possible for University Laboratory High School students.

Thank you for listening.

References

Lean resources:

- https://leanprover-community.github.io/

Euclid's axioms:

- David E. Joyce. *Euclid's Elements, Book 1.* Clark University, Worchester, MA 01610
- Euclidean and Non-Euclidean Geometries: Development and History (Book, Chapter 1), Marvin J. Greenberg.

Hilbert's axioms:

- Gabriel Braun, Julien Narboux. *From Tarski to Hilbert.* Automated Deduction in Geometry 2012, Jacques Fleuriot, Sep 2012, Edinburgh, United Kingdom. pp.89-109, ff10.1007/978-3-642-40672-0_7ff.
- K. Borsuk and Wanda Szmielew. (1960). *Foundations of geometry, Euclidean and Bolyai-Lobachevskian geometry: projective geometry.*
- E. J. Townsend, translator. *The Foundations of Geometry*. By David Hilbert, The Open Court Publishing Company, 1950