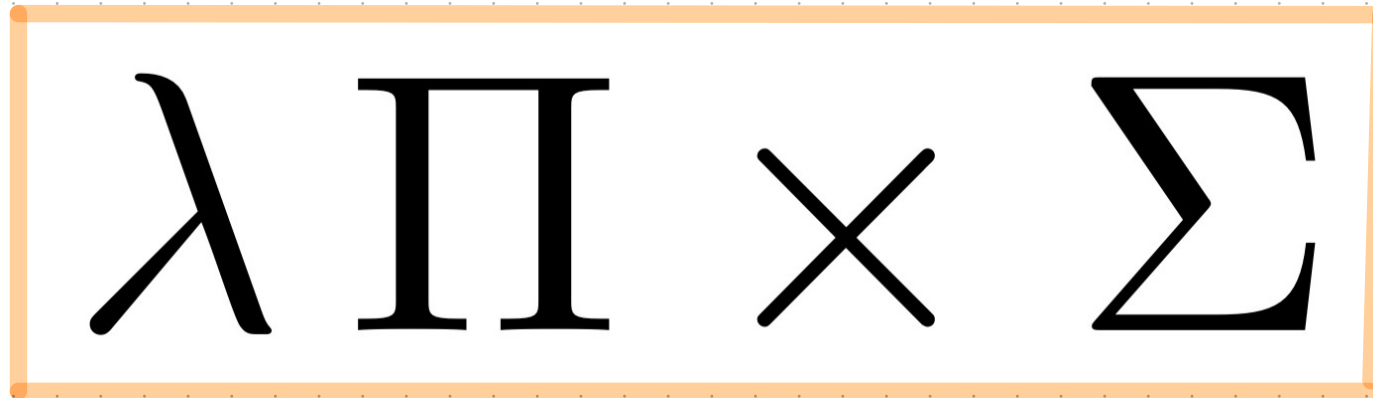


HEGL Illustrating Mathematics Seminar  
Winter Semester 2024-2025



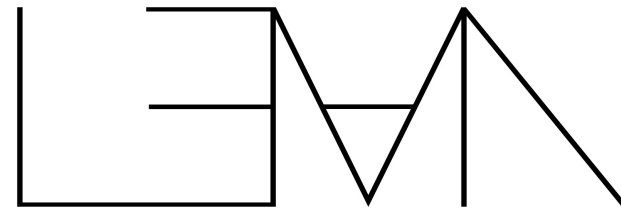
Logic, algebra and proof visualization

Florent Schaffhauser  
Heidelberg University

# What is a mathematical proof and how can we represent it?

- In this seminar, we take a **constructive approach** to this question.
- Proving a theorem means constructing a proof of it, much like when you introduce a term of a certain type in a **programming language**.
- The underlying logical foundations of this approach make **functions** the primitive object, from which everything else is defined.

Can we use this to venture a guess as to how mathematics will be taught at the university level 50 years from now?



## Programming Language and Theorem Prover

```
mathlib4
SumsOfSquares.lean x Defs.lean
Mathlib > Algebra > Ring > SumsOfSquares.lean > isSumSq.add
40 variable {R : Type*}
41
42 /--
43 In a type `R` with an addition, a zero element and a multiplication, the property of being a sum of
44 squares is defined by an inductive predicate: `0 : R` is a sum of squares and if `S` is a sum of
45 squares, then for all `a : R`, `a * a + S` is a sum of squares in `R`.
46 -/
47
48 @[mk_iff]
49 inductive isSumSq [Add R] [Zero R] [Mul R] : R → Prop
50 | zero : isSumSq 0
51 | sq_add (a S : R) (pS : isSumSq S) : isSumSq (a * a + S)
52
53 /--
54 If `S1` and `S2` are sums of squares in a semiring `R`, then `S1 + S2` is a sum of squares in `R`.
55 -/
56
57 theorem isSumSq.add [AddMonoid R] [Mul R] {S1 S2 : R} (p1 : isSumSq S1)
58 | (p2 : isSumSq S2) : isSumSq (S1 + S2) := by
59   induction p1 with
60 | zero => rewrite [zero_add]; exact p2
61 | sq_add a S pS ih => rewrite [add_assoc]; exact isSumSq.sq_add a (S + S2) ih
62
63 variable (R) in
64
65 /--
66 In an additive monoid with multiplication `R`, the type `SumSqIn R` is the submonoid of sums of
67 squares in `R`.
68 -/
69
70 def SumSqIn [AddMonoid R] [Mul R] : AddSubmonoid R where
71   carrier := {S : R | isSumSq S}
72   zero_mem' := isSumSq.zero
73   add_mem' := isSumSq.add
74
```

Lean Infoview

- ▼ SumsOfSquares.lean:60:52
  - ▼ Tactic state
    - No goals
  - ▼ Expected type
    - R : Type u\_1
    - inst! : AddMonoid R
    - inst! : Mul R
    - S1 S2 : R
    - p2 : isSumSq S2
    - ┆ isSumSq S2
  - ▶ All Messages (0)

Restart File

matematiflo-semireal-rings-defs 1 0 0 1 0 Pull Request #14941 Ln 60, Col 53 Spaces: 2 UTF-8 LF lean4

## Constructive mathematics

---

The constructive approach to mathematics is often reduced (quite misleadingly) to mathematics *without* the law of excluded middle (LEM), or without the axiom of choice (AC). It should instead be thought of as a sum of algorithmic procedures for deriving proofs, which *may or may not* use LEM or AC.

Conceptually, this is made possible by an interpretation of mathematical statements in a way that lays the emphasis on computation. Existential statements, in particular, should be formally interpreted in a constructive manner (which calls for an explicit construction of an element satisfying a certain property).

Additionally, logic is not a pre-requisite in this approach. Instead, logical connectives and quantifiers are introduced formally in constructive mathematics, using the same language used to introduce mathematical objects.

## List of topics for the talks

---

- Sets and logic.
- Basic algebra.
- Rings and modules.
- Divisibility in discrete domains.
- Principal ideal domains.

In the seminar, we will focus on undergraduate commutative algebra: rings, fields and algebras over a field.

As a first step, participants will give a presentation on a subject of their choice from the list of topics proposed by the coordinator.

This talk, and its corresponding handout, will count towards 50% of the final grade.

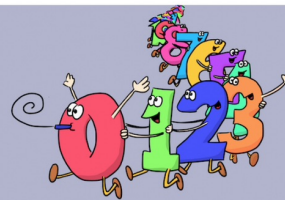
# Lean Game Server

A repository of learning games for the proof assistant Lean  
(Lean 4) and its mathematical library mathlib

In the second step,  
we will work  
collaboratively to  
elaborate an **online**  
**tutorial** to some of  
the mathematical  
results presented in  
the first part.

## Natural Number Game

The classical introduction game for Lean.

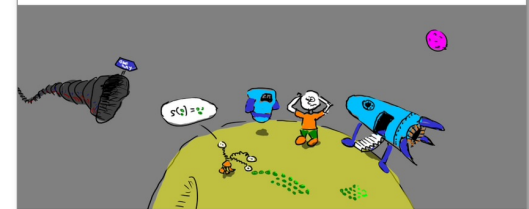


In this game you recreate the natural numbers  $\mathbb{N}$  from the Peano axioms, learning the basics about theorem proving in Lean.

This is a good first introduction to Lean!

## Robo

Erkunde das Leansche Universum mit deinem Robo, welcher dir bei der Verständigung mit den Formalosophen zur Seite steht.



Dieses Spiel führt die Grundlagen zur Beweisführung in Lean ein und schneidet danach verschiedene Bereiche des Bachelorstudiums an.

(Das Spiel befindet sich noch in der Entstehungsphase.)

The ideal output is a kind of "Heidelberg Lean game"  
that can be played online by future generations of  
students, to teach themselves the basics of type-  
theoretic mathematics.

# Paperproof

## A new proof interface for Lean 4.

The screenshot shows the Paperproof interface with a Lean 4 proof script on the left and a visual proof diagram on the right. The script is as follows:

```
1 import Mathlib.Data.Set.Basic
2 import Paperproof
3
4 theorem commutativityOfIntersections
5 (s t : Set Nat) : s ∩ t = t ∩ s := by
6   ext x
7   apply Iff.intro
8
9   --intro h1
10  rw [Set.mem_inter_iff, and_comm] at h1
11  exact h1
12
13  --intro h2
14  rw [Set.mem_inter_iff, and_comm] at h2
15  exact h2
16
```

The visual proof diagram on the right illustrates the proof steps. It starts with the goal  $s \cap t = t \cap s$  and the variable  $x$ . The proof is split into two paths,  $h1$  and  $h2$ , which are symmetric. For  $h1$ , the goal is  $x \in s \cap t$ , which is rewritten to  $x \in s \wedge x \in t$  using `rw [Set.mem_inter_iff]`, and then to  $x \in t \wedge x \in s$  using `rw [and_comm]`. The goal is then solved by `exact h1`. A similar process occurs for  $h2$ . The two paths are then combined using `apply Iff.intro` to reach the final goal  $s \cap t = t \cap s$ .

Other proof visualization techniques such as Paperproof or Verbose might be explored too.

The final presentation and HEGLE blog post, by the students, will count towards the remaining 50% of the grade.

The screenshot shows the Verbose interface with a Lean 4 proof script on the left and a visual proof diagram on the right. The script is as follows:

```
import Verbose.English.ExampleLib
import Verbose.English.Statements

set_option verbose.suggestion_widget true

Exercise "Continuity implies sequential continuity" declaration uses 'sorry'
Given: (f : ℝ → ℝ) (u : ℕ → ℝ) (x₀ : ℝ)
Assume: (hu : u converges to x₀) (hf : f is continuous at x₀)
Conclusion: (f ∘ u) converges to f x₀

Proof:
Let's prove that  $\forall \epsilon > 0, \exists N, \forall n \geq N, |(f \circ u) n - f x_0| \leq \epsilon$ 
Fix  $\epsilon > 0$ 
By hf applied to  $\epsilon$  using that  $\epsilon > 0$  we get  $\delta$  such that  $(\delta\_pos : \delta > 0) (h\delta : \forall (x : \mathbb{R}), |x - x_0| \leq \delta \rightarrow |f x - f x_0| \leq \epsilon)$ 
By hu applied to  $\delta$  using that  $\delta > 0$  we get  $N$  such that  $hN : \forall n \geq N, |u n - x_0| \leq \delta$ 
Let's prove that  $N$  works:  $\forall n \geq N, |(f \circ u) n - f x_0| \leq \epsilon$ 
Fix  $n \geq N$ 
By hN applied to  $n$  using that  $n \geq N$  we get  $H : |u n - x_0| \leq \delta$ 

sorry
QED
```

The visual proof diagram on the right shows the tactic state and the goal. The goal is  $(f \circ u) n - f x_0 \leq \epsilon$ . The diagram shows the current state of the proof, including the goal, the current state, and the suggestions provided by the interface.