Theorem proving in Lean

Matúš Behun

Slovak University of Technology in Bratislava

March 17, 2021

Overview

Computer aided proofs

Proving mathematical statements, showing that assumptions lead to the conclusion with help of automation.

Four color theorem

Plane divided into regions can be colored using 4-colors in such a way that no boundary share same color.

After many attempts proved (1976) partially with help of computer. 1946 configurations were checked by computer.

Proof assistants

Software which requires human interaction during the process. Output has to be readable for human to make decision what to do next.

- Lean(2013)
- HOL - Higher Order Logic(1988)
- Coq(1989)

Why should we use proof assistants?

- Some mathematical fields more prone to make mistake
- Shinichi Mochizuki proving abc conjecture

Curry-Howard correspondence

Establishes relation between forumlas and proofs of those formulas in propositional intuitionistic logic and functions of a given type in a functional programming language.

First order logic formulas

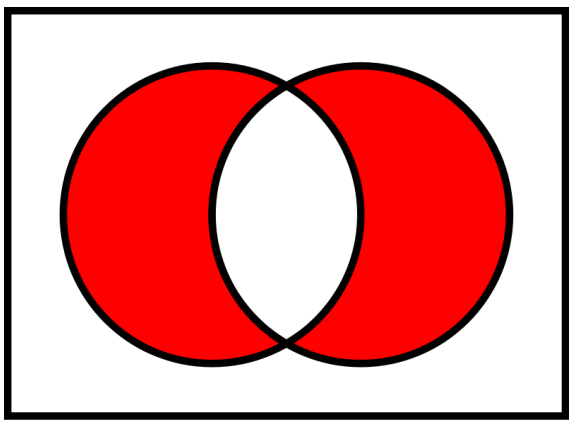
- Variables called Terms
- Relations $\Rightarrow, \wedge, \vee, \ldots$

Sets

- Arbitrary elements

Curry-howard correspondence

Predicate logic	Sets
$p \wedge q$	$P \times Q = \{(p, q) \mid p \in P \text{ and } q \in Q\}$
$p \vee q$	$P \oplus Q = \{x \mid x \in P \vee x \in Q \text{ and } x \notin P \wedge x \notin Q$



Curry-howard correspondence

Predicate logic

- $p \Rightarrow q$
- $\neg p$

p	q	$p \Rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Sets

- $P^Q \Leftrightarrow [P, Q]$
- $P^\emptyset$

Curry-howard correspondence

Predicate logic	Sets
$p \wedge q \Rightarrow q$	$[P \times Q, Q]$

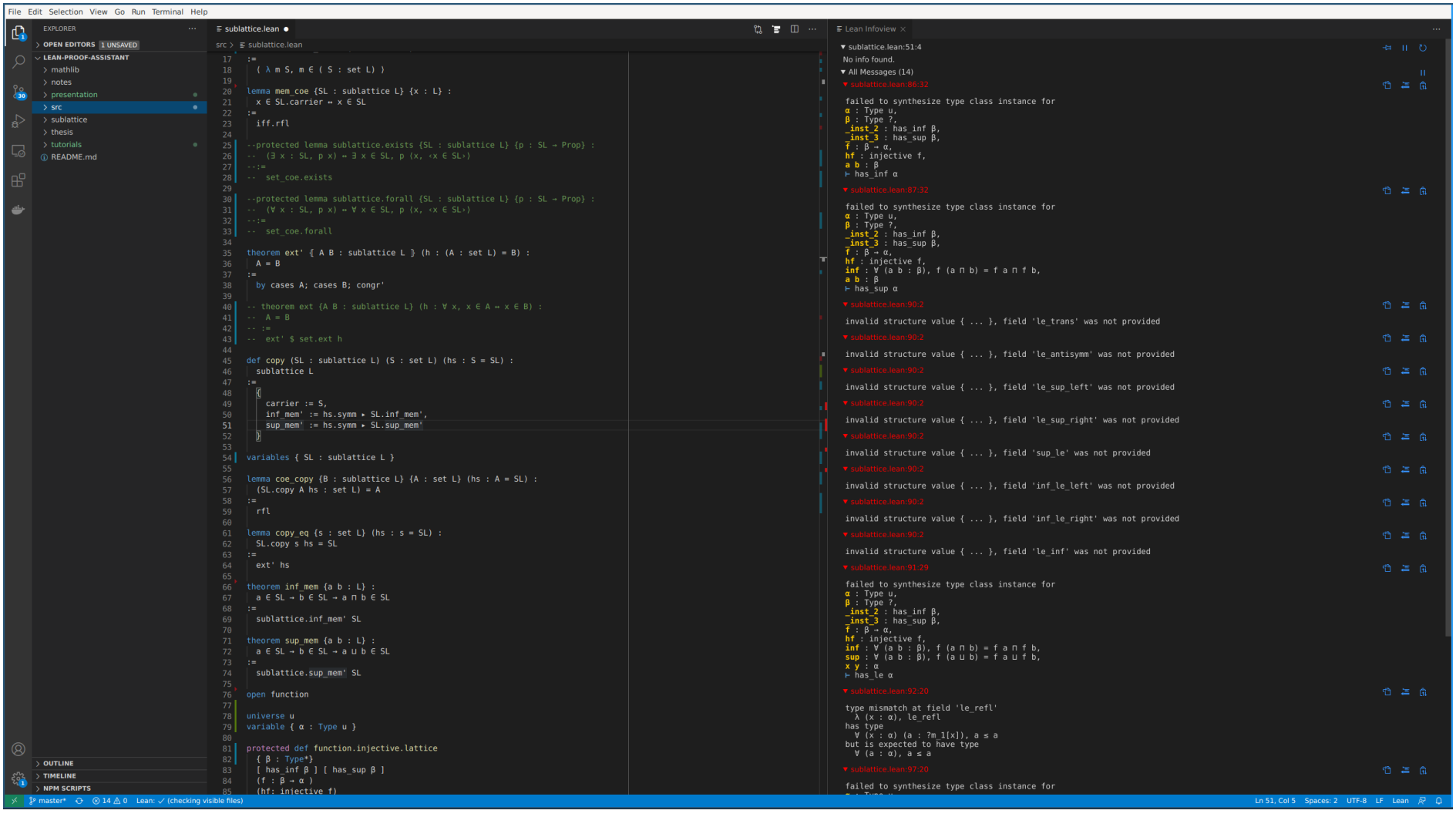
Lean theorem prover

- Functional programming language
- Interactive theorem prover
- Open source project backed by Microsoft Research

Why Lean

- Able to make sophisticated objects - Perfectoid spaces
- used by XENA project, get mathematicians to use proof verification software
- mathlib(volunteering library of mathematics)
- active community

Lean enviroment



Forward proving

Two types of proving in Lean forward, backwards

```
variables p q :  Prop

theorem t1 :
p → q → p
:=
λ hp :  p,
λ hq :  q,
hp
```

Backwards proving

```
variables p q :  Prop

example Q → (P ∨ Q)
:=
intro a,
right
```

Backwards proving

```
import algebra.group.defs
variables (G : Type) [group G] (a b c :  G)
example :  a * a⁻¹ * 1 * b = b * c * c⁻¹
:=
begin
simp
end
```