数学形式化证明与定理证明器 Lean4 简介

郭俊余（Hagb）

hagb@hagb.name
https://github.com/Hagb
该幻灯片及用到的代码见于 https://hagb.name/lean4-aosc

2025-11-22

中山大學
SUN YAT-SEN UNIVERSITY

一个例子

一个例子

例子 (另见第 18 页)

假设 x 为实数，那么 $x^3+x^2+x+1=0$ 当且仅当 $x=1$ 或 $x=-1$

“证明”.

由 $x^3+x^2+x+1=0$ 易得 $x\neq 0$，从而可得 $x^2+x+1+\frac{1}{x}=0$ 即 $x^2+x+1=-\frac{1}{x}$，代入原式可得

$$x^3+x^2+x+1=0 \Leftrightarrow x^3-\frac{1}{x}=0 \Leftrightarrow x^4-1=0 \Leftrightarrow x=1 \vee x=-1.$$

□

“推论”

令 $x=1$，于是 $1^3+1^2+1+1=0$，即 $4=0$.

修改自知乎用户 Archimon 的回答 https://www.zhihu.com/question/1892270149861617744/answer/1903571259041775776.

非形式证明与形式证明

非形式证明与形式证明

	非形式证明	形式证明
语言	自然语言和数学语言	精确定义的形式语言
推理方式	混合数学直觉和逻辑推理、省略细节	严格基于逻辑和公理
可读性	易写、易读，便于交流	人类读写较为困难
有效性验证	人工审核，无法机械化验证	可以机械化进行验证
可靠性	可能有较难以察觉漏洞	没有漏洞

Table 1: 非形式证明 vs 形式证明

定理证明器的工作流程

形式化证明代码

- 更接近自然语言
- 自动化
- 针对人类设计

↓ 繁饰（Elaboration）

逻辑表达式

- 简单 & 精确
- 形式证明
- 为计算机设计

↓ 验证（Verification）

- 检查形式证明
- 小量代码
- 对证明正确性最关键的部分

定理证明器（不）是什么

是什么/做什么：

- 机械化地验证用户输入的证明代码
- 提供有限的证明自动化
- 仅让真命题能通过验证
- ~~把计算机变成暖风机~~

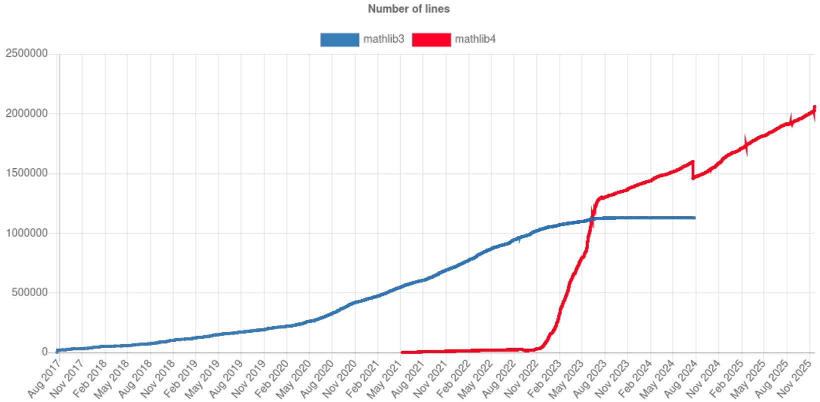
不是什么/**不**做什么：

- 不是 AI
- 不是计算机代数系统
- 不能自动发现和/或证明非平凡的定理
- 不能以人类友好的方式指出伪证的错误

Lean 定理证明器

Mathlib——Lean 的数学库

- 包含多个数学分支的形式化[1]
- 共有 120216 个定义、242245 个定理、653 个贡献者[2]



[1] https://leanprover-community.github.io/mathlib-overview.html
[2] 截至 2025-11-21，https://leanprover-community.github.io/mathlib_stats.html

例子 1——Lean 的类型

```
#check 0             -- 输出 0 : ℕ
#check 0+1=1         -- 输出 0 + 1 = 1 : Prop
#check 0+1=2         -- 输出 0 + 1 = 2 : Prop
#check zero_add 1    -- 输出 zero_add 1 : 0 + 1 = 1
```

例子 2——Lean 中的简单证明与证明策略 I

```lean
import Mathlib

-- “若 a = 0，那么称 eq_zero a”
def eq_zero (a : ℕ) : Prop := a = 0

-- 下面给出 0 = 0 和 0 + 1 = 1 这两个命题的证明，
lemma zero_eq_zero : eq_zero 0 := refl 0
lemma zero_add_one_eq_one : 0 + 1 = 1 := zero_add 1

example : 0 = 0 := refl 0

lemma le_of_le_of_le {a b c d: ℕ}
    (h₁ : a ≤ b) (h_eq : b = c) (h₂ : c ≤ d) :
    a ≤ d :=
  le_trans (le_of_le_of_eq h₁ h_eq) h₂
```

例子 2——Lean 中的简单证明与证明策略 II

更接近于非形式证明的证明:

```
lemma le_of_le_of_le'' {a b c d: ℕ}
    (h₁ : a ≤ b) (h_eq : b = c) (h₂ : c ≤ d) :
    a ≤ d := by
  rw [h_eq] at h₁
  exact le_trans h₁ h₂

lemma le_of_le_of_le_of_eq'' {a b c d: ℕ}
    (h₁ : a ≤ b) (h₂ : b ≤ c) (h₃ : c = d) :
    a ≤ d := by
  linarith
```

例子 3——整除关系

定义

```
def mydvd (a : ℕ) (b : ℕ) := ∃ x : ℕ, b = a * x
infix:50 " ∣' " => mydvd
```

传递性

```
theorem mydvd_trans :
    a ∣' b → b ∣' c → a ∣' c := by
  intro h₁ h₂
  cases' h₁ with x₁ h₁
  cases' h₂ with x₂ h₂
  use x₁ * x₂
  rw [h₂, h₁]
  ring
```

例子 4——Lean 中的多态 I

考虑以下命题

命题的原始叙述

如果 f 是线性的，那么 $f(3 \cdot x + y) = 3 \cdot f(x) + f(y)$.

例子 4——Lean 中的多态 II

命题稍微详细的叙述

若 K 是一个域，U 和 V 是 K 上的向量空间，那么如果 $f : U \to V$ 是线性的，则 $\forall x, y \in U, f(3 \cdot x + y) = 3 \cdot f(x) + f(y)$.

例子 4——Lean 中的多态 III

命题详尽的叙述

若

- K 装备了加法 $+_K$ 和乘法 $\cdot_K$ 后为一个域
- U 上装备了加法 $+_U$ 和关于域 K' 的标量乘法 $\cdot_{K,U}$ 则为向量空间 U'
- V 装备了加法 $+_V$ 和关于域 K' 有标量乘法 $\cdot_{K,V}$ 则为向量空间 V'

那么如果 $f : U \to V$ 为向量空间 U' 到 V' 的线性映射，则
$\forall x, y \in U, f(3_K \cdot_{K,U} x +_U y) = 3_K \cdot_{K,V} f(x) +_V f(y).$

例子 4——Lean 中的多态 IV

Lean 形式化代码

```
import Mathlib
variable (U : Type*)          -- U 是任意类型
variable [AddCommGroup U]  -- U 有加法交换群结构
variable (V : Type*) [AddCommGroup V]  -- 同上
variable (K : Type*) [Field K] -- K 有域结构
-- K 关于 U、V 有线性的标量乘法
variable [Module K V] [Module K U]
-- 假设 f 为 U 到 V 关于 K 的线性映射
variable (f : U →ₗ[K] V)

example: ∀ x y, f (2•x + y) = 2•f x + f y :=
  by simp
```

中山大學 SUN YAT-SEN UNIVERSITY

回顾 $4 = 0$ 的伪证

非形式伪证于第 3 页

```
lemma a_wrong_proof (x : ℝ) :
    x ^ 3 + x ^ 2 + x + 1 = 0 ↔ x = 1 ∨ x = -1 := by
  trans x ^ 3 - 1 / x = 0 ∧ x ≠ 0
  · -- 此处待证: x ^ 3 + x ^ 2 + x + 1 ↔ (x ^ 3 - 1 / x = 0 ∧ x ≠ 0)
    sorry
  trans x ^ 4 - 1 = 0
  · grind -- 此处证: (x ^ 3 - 1 / x = 0 ∧ x ≠ 0) ↔ x ^ 4 - 1 = 0
  -- 之后需要证 x ^ 4 - 1 = 0 ↔ x = 1 ∨ x = -1
  have factor : x^4 - 1 = (x-1) * (x+1) * (x^2 + 1) := by grind
  have not_vanish : x ^ 2 + 1 ≠ 0 := by nlinarith
  simp [factor, not_vanish, sub_eq_zero, add_eq_zero_iff_eq_neg]

lemma four_eq_zero : (4 : ℝ) = 0 := by
  convert (example1 1).mpr (by left; rfl)
  norm_num
```

谢谢大家

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

此外许多资料亦有中文翻译：https://www.leanprover.cn