12.3 Claude自动形式化总结

1. 251 in zihan

Try1: 小错,

example {G : Type\*} [Group G] {g : G} (h : g ^ 2 = 1) : g = g⁻¹ := by

-- Start with g^2 = 1 and multiply both sides by g^{-1} on the left

have h1 : g⁻¹ \* (g \* g) = g⁻¹ \* 1 := by

rw [← pow\_two] -- (1) g ^2 and g \* g are different

rw [h]

-- Use associativity and simplify right side

rw [← mul\_assoc, mul\_one] at h1 -- (2) mul\_assoc direction

-- Simplify left side using g^{-1} \* g = 1

rw [inv\_mul\_self, one\_mul] at h1

-- h1 now shows g = g^{-1}

exact h1.symm -- (3) after fix (1) (2), do not need this line

总结: 小错 跳步, 方向不对

1. 253 in zihan

import Mathlib

example {G : Type\*} [Group G] (a b : G) (h : a \* b = b \* a⁻¹) : b \* a = a⁻¹ \* b := by

-- multiply a on both sides of h to get a new equation

have h1 : a \* b \* a = b \* a⁻¹ \* a := by exact mul\_right\_cancel\_iff.mpr h

-- simplify h1 by cancelling a⁻¹ \* a

rw [inv\_mul\_cancel\_right] at h1

-- use h1 to prove the goal through a series of rewrites

rw [← h1, ← mul\_assoc, ← mul\_assoc, inv\_mul\_self, one\_mul, h, mul\_assoc b, inv\_mul\_self, mul\_one]

完全正确

-- 能写出长rw

1. 326 in zihan

四个方向错误 a \* b, b\*a (两次), 1 \* a , a\*1(两次)其他都对

1. 251

根本没有形式化,

This formal proof follows the structure of the informal proof closely:

1. We define the polynomials `f` and `f₁` as specified.

2. We prove that the degree of `f` equals the cardinality of F and thus is non-zero.

3. We show that `f` and `f₁` have the same degree using `natDegree\_add\_eq\_left\_of\_natDegree\_lt`.

4. We deduce that `f₁` has non-zero degree.

5. Using algebraic closure, we obtain a root of `f₁`.

6. We derive the contradiction by showing that if `x` is a root of `f₁`, then `f₁(x) = f(x) + 1 = 1`, which contradicts `f₁(x) = 0`.

给例子应该能解决不形式化的问题

1. 253

Import问题

只缺了一步symm其他都对

-- Use x \* y = 1 to show y = z

calc z = z \* (x \* y) := by rw [h1]; rw [mul\_one]

\_ = (z \* x) \* y := by rw [mul\_assoc]

\_ = 1 \* y := by rw [h2]

\_ = y := by rw [one\_mul]

说了要证y = z,还是证了 z = y

1. 326

Import 自己编 还import mil

have h\_f\_eval : f.eval x = 0 := by

simp [f] at hx h\_eval ⊢

rw [h\_eval] at hx

exact sub\_eq\_zero.1 hx

--- 没有自然语言导引，选择了错误的做法

大量的类型问题 ≠ 0 和 > 0 的混用, natDegree 和degree的转换问题

1. 没import

遵循自然语言导引,自己编定理.

.Characteristic 是class, 不能直接用

-- First, rewrite p^(d+1) as p \* p^d

have h1 : p ^ (d + 1) = p \* p ^ d := by

simp [pow\_succ]

差一个symm

差mul\_comm, pow\_mul 方向不对

-- Now we can use the Frobenius endomorphism in characteristic p

-- (a + b)^p = a^p + b^p in characteristic p

rw [frobenius\_add pchar (x^(p^d)) (y^(p^d))]

-- 定理相似，但是不完全一样, 需要先rw [frobenius\_def]

(这里erw也不行,不知道为什么).

最后一行可以去掉

Claude:

1. 精确程度收到自然语言影响, 自然语言里等同的,但在形式化里不等同的,会被当成等同 “g ^ 2” “g \* g”, natDegree degree, ≠ 0 和 > 0. 还有 x ^ p和frobenius R p

让大模型把自然语言细细切做臊子,表现能否有提升?

1. 在没有自然语言导引但是还真有点数学的地方,可能会因为不懂数学而瞎写. (参考253)
2. 喜欢自己编import毫无意义,还喜欢import MIL.common
3. 有很多2择1的事情 选择错误
4. 自然语言里的断言, 大模型有时会觉得这应该在mathlib里有,从而开始猜名字 (prompt + RAG lean search也许能解决,没搜到就是没有) 不仅需要提供定理名,还需要提供定义/定理具体内容
5. 令人惊讶的成功生成了长rw