生成AI×定理証明支援系で変える数学証明問題の採点

MS QULTIVA Course C 成果物発表

鈴木 瞭賀

目次

①Cコースとは何か
— テーマ設定と取り組みの特徴

②成果物の紹介
— 生成AI × 定理証明支援系による
数学証明問題の自動採点

③学びと今後の展望
— 視点の変化とこれからの活用

①Cコースとは何か
― テーマ設定と取り組みの特徴

MS QULTIVA Cコースとは

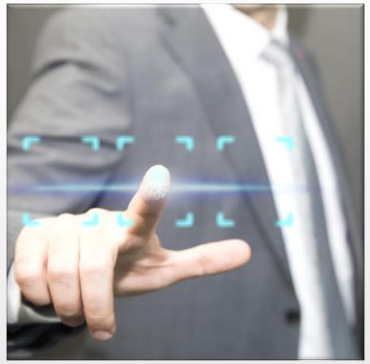
- 技術やAIを使い、自分で課題を設定し形にするコース
- 正解や完成形が最初から決まっていない
- 自分の独自テーマを軸に進行
- MS Baseさんに相談しながら方向性を整理

②成果物の紹介
― 生成AI × 定理証明支援系による数学証明問題の自動採点

課題設定・観察

近年、社会的な課題として注目されている教師の負担の大きさ

　授業以外にも、採点や記録作業など多くの業務を担っており、特に数学の証明問題のような記述式答案は、一つ一つ確認する必要があるため採点に時間がかかる。その結果、教師の負担が増え、生徒への十分なフィードバックが難しくなっている。



記述式問題の採点における、採点基準の不公平さ

　地域や学校、担当する教師によって評価の基準や厳しさが異なると、同じ内容の解答であっても結果に差が生じる可能性がある。特に入試や内申点などの重要な評価場面では、こうしたばらつきが不公平感につながる恐れがある。

自学習の場面では、採点者がいないことが大きな課題となる

　記述式問題では、自分で正誤を判断することが難しく、自己採点では理解に不安が残りやすい。一方、採点者を用意するには人件費や時間がかかり、その場でフィードバックを受けることが難しい。

これらの課題をどう解決するか

求められる解決の方向性

- 教師の負担を増やさずに採点できること
- 採点基準が人や環境によってぶれないこと
- 自学習でも、すぐにフィードバックが得られること

↓

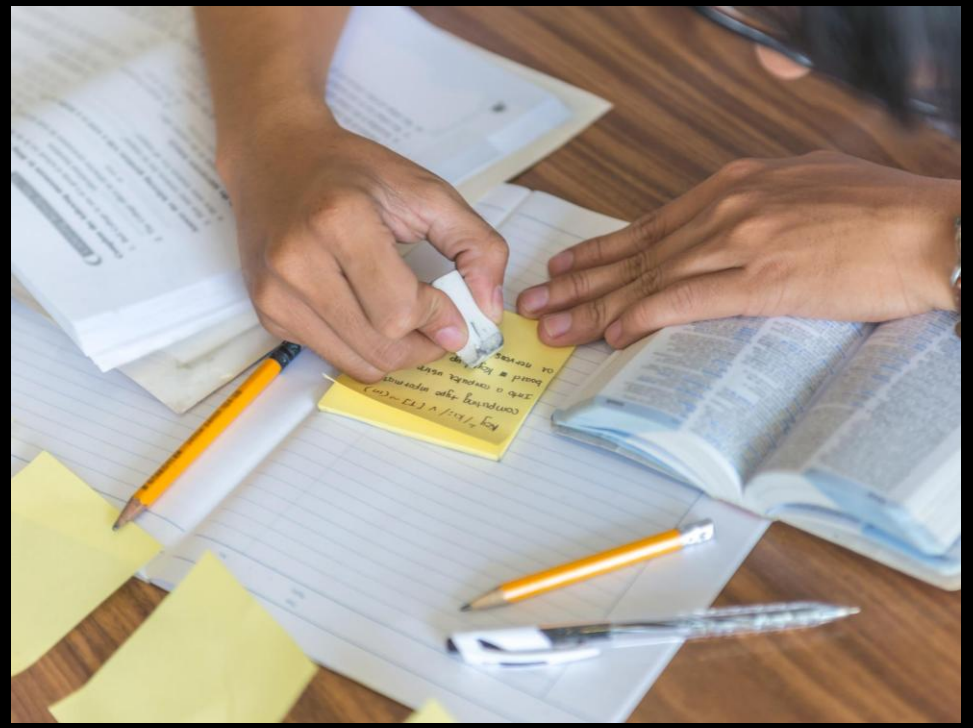
必要となる仕組み

- 記述式の証明を自動で理解・評価できる
- 人の主観に依らず、論理的に正しさを判定できる
- いつでもどこでも人に頼らず、採点できる

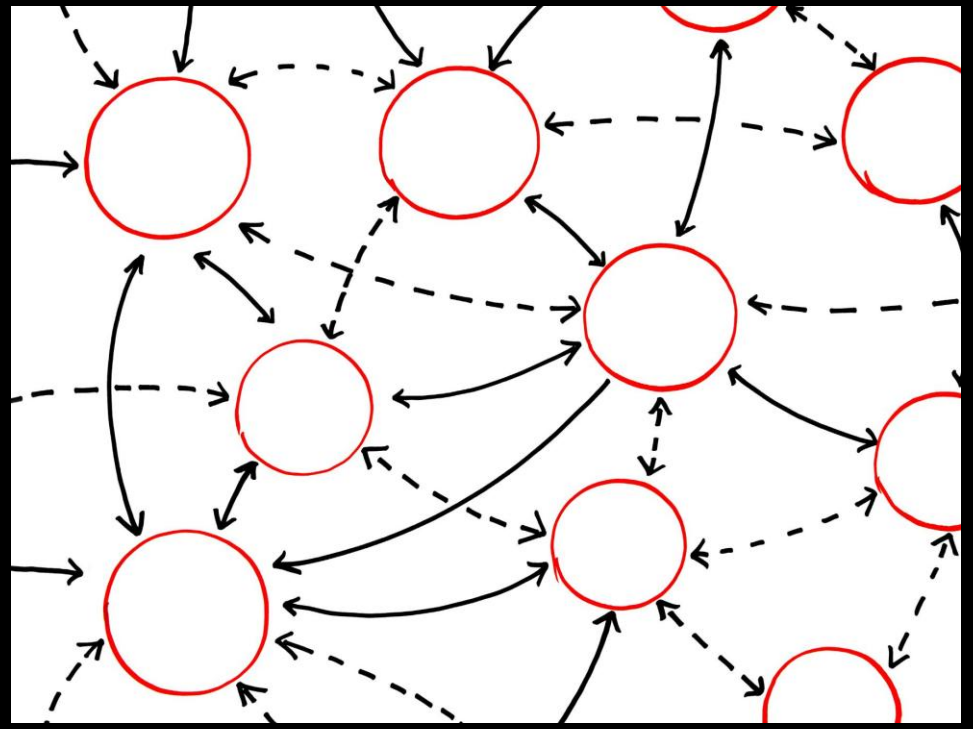
↓

解決手段

- **生成AI × 定理証明支援系による数学証明問題の自動採点**



生成AI×定理証明支援系で数学証明問題の自動採点を実現



- 数学の「証明」は、答えだけでなく**論理の流れ**が重要
- 生成AIを使い、**人が書いた証明をAIが理解・整形**
- 定理証明支援系「**Lean 4**」を用いて
  → 論理が正しいかを**コンピュータで厳密に検証**
- **AI：柔軟な文章理解**
- **Lean 4：厳密で公平な正誤判定**
- 両者を組み合わせることで
  👉 **公平・高速・自動の証明採点**を実現

数学証明 採点システム

問題文

日本語解答

採点する

判定された証明タイプ

未判定

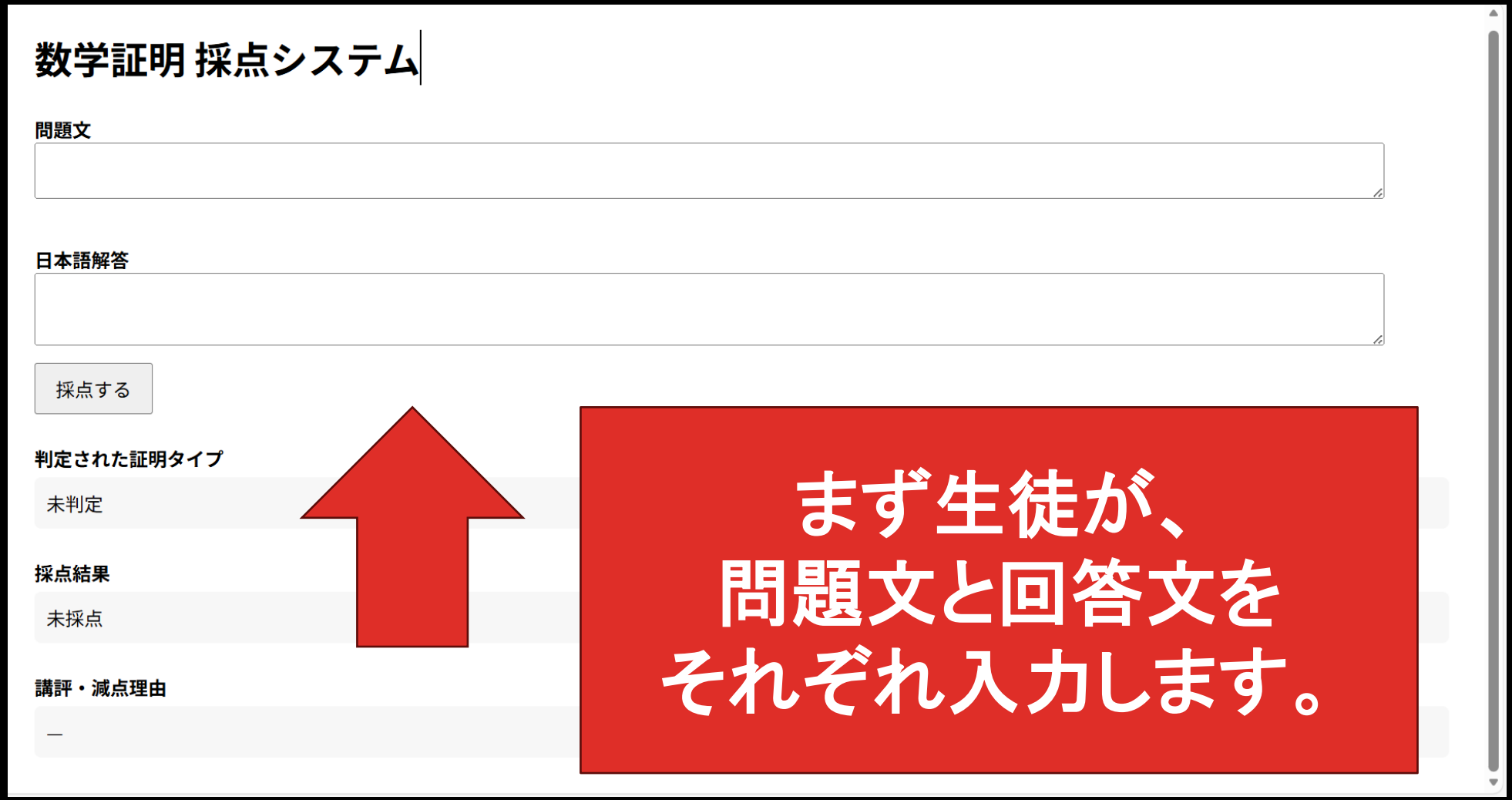
採点結果

未採点

講評・減点理由

—

自動採点ツールの全体構成

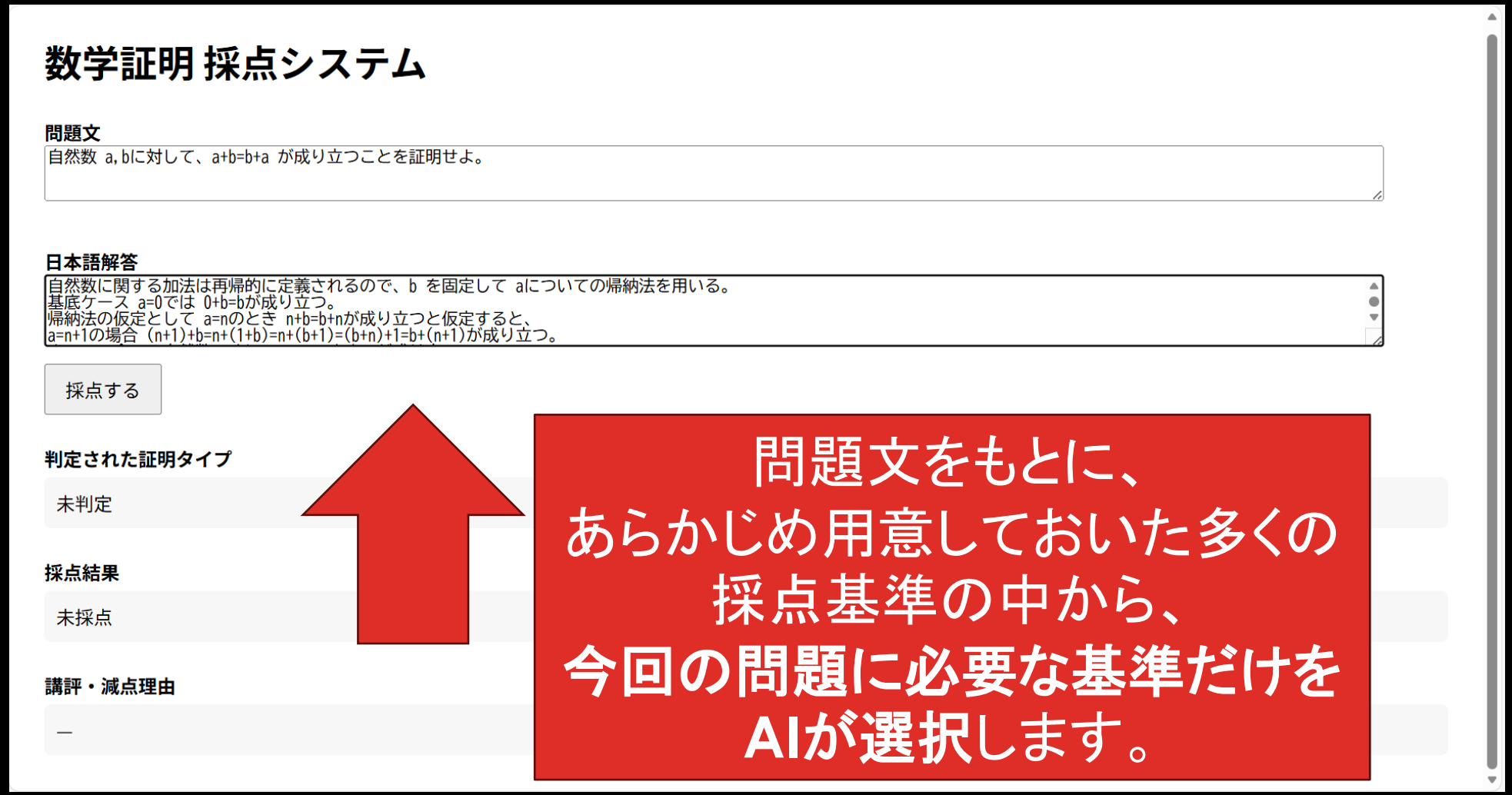


自動採点ツールの全体構成

```
C:\math-grader>python test_lean.py
===== 生成された Lean コード =====
lemma add_zero (a : Nat) : a + 0 = a := by
  induction a with
  | zero => rfl
  | succ n ih =>
    calc
      succ n + 0 = succ (n + 0) := rfl
      _ = succ n := by rw [ih]

lemma add_succ (a b : Nat) : a + succ b = succ (a + b) := by
  induction a with
  | zero => rfl
  | succ n ih =>
    calc
      succ n + succ b
        = succ (n + succ b)   := rfl
      _ = succ (succ (n + b)) := by rw [ih]
      _ = succ (succ n + b)   := rfl

theorem add_comm (a b : Nat) : a + b = b + a := by
  induction a generalizing b with
  | zero =>
    calc
      0 + b = b := rfl
      _ = b + 0 := by rw [add_zero b]
  | succ n ih =>
    calc
      (succ n) + b
        = succ (n + b)    := rfl
      _ = succ (b + n)    := by rw [ih]
      _ = b + succ n      := by rw [add_succ b n]

C:\math-grader>python test_lean.py
===== 生成された Lean コード =====
open Nat
```

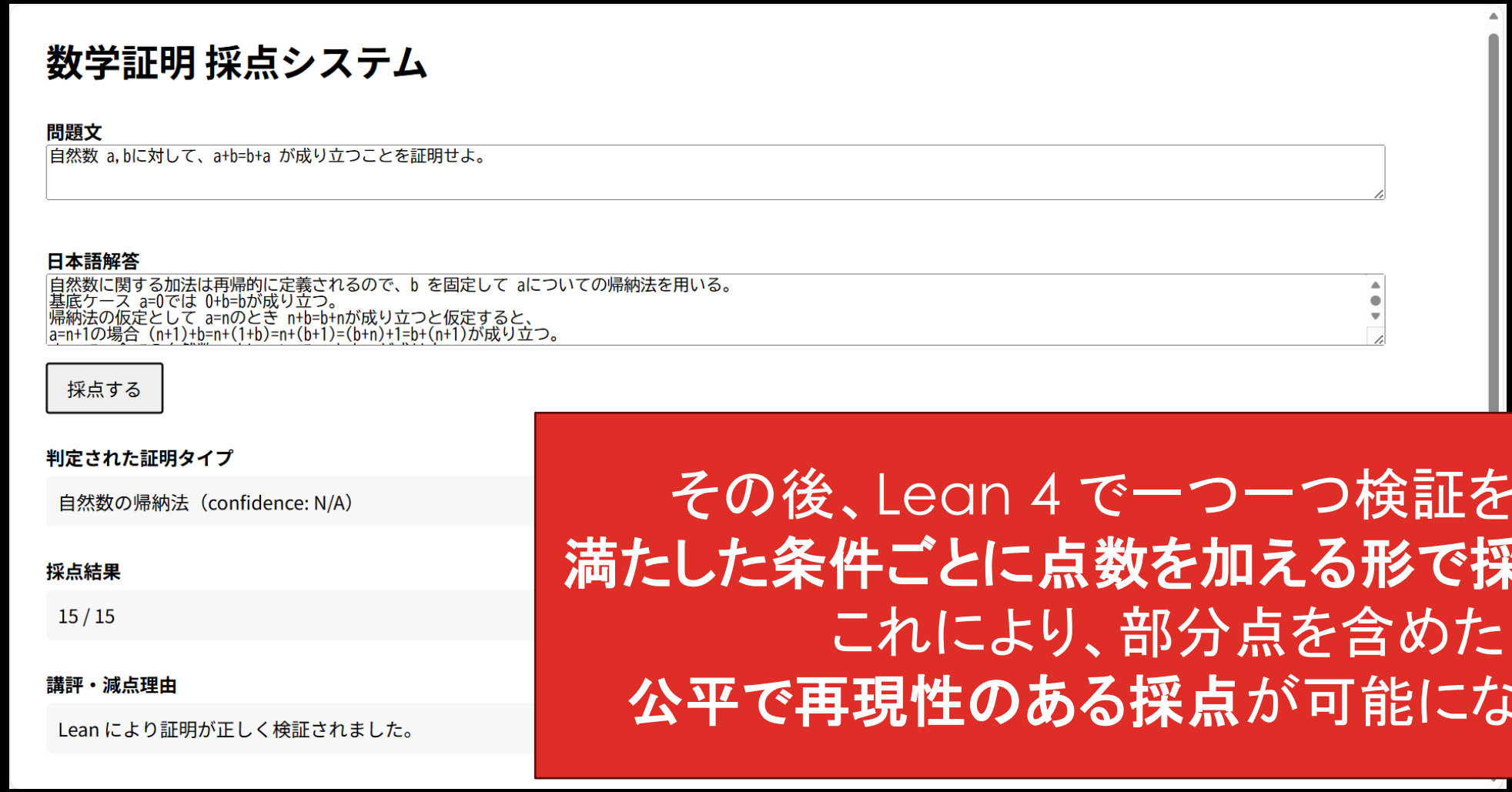
それをツール側で
Gemini-2.5-flash を
API 経由で呼び出し、
数学的な内容を Lean 4 の
コードに翻訳します。

自動採点ツールの全体構成



問題文をもとに、
あらかじめ用意しておいた多くの
採点基準の中から、
今回の問題に必要な基準だけを
AIが選択します。

自動採点ツールの全体構成



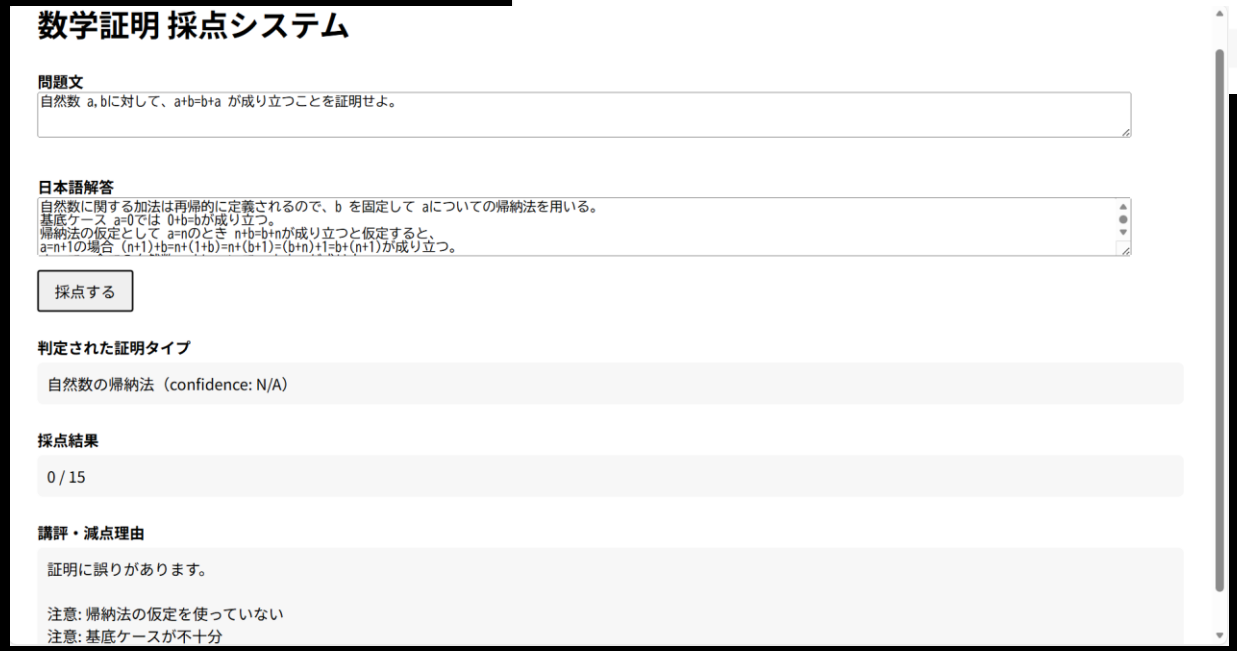
その後、Lean 4 で一つ一つ検証を行い、
満たした条件ごとに点数を加える形で採点します。
これにより、部分点を含めた
公平で再現性のある採点が可能になります。

結果：課題①：AIによる不公正性の可能性

- 生成AIは確率的に文章やコードを生成する
- 同じ入力でも、**生成結果が変わる可能性**がある

影響

- 翻訳されたLeanコードの違いにより
  - ▶ 採点結果が変動するリスク
- 採点の**再現性・公平性**に課題

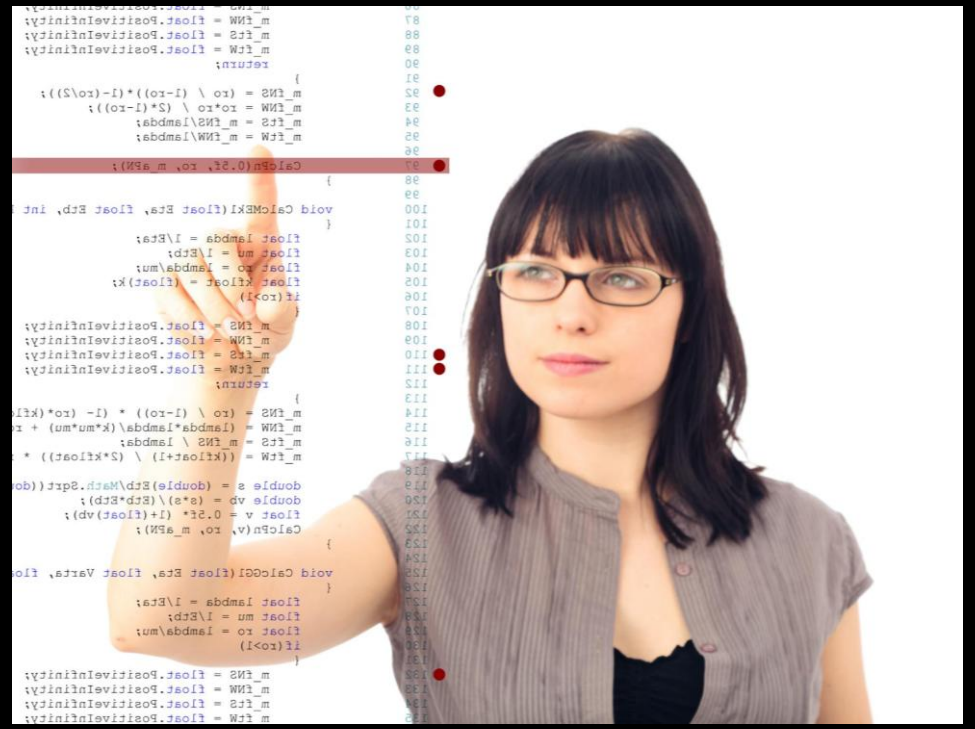


結果：課題②：多様な証明アプローチへの未対応

- 数学の証明には
  - ▶ 方針・構成・補題の置き方など**多様な書き方**がある
- 現在は
  - ▶ 想定された構造の証明に強く依存

影響

- ▶ 同じ問題でも複数の証明方法が存在するケースに対応できるようにする必要がある



MSさんとの連携を通じた学びと考え方の変化

MS Baseさんとの議論からの学び

- AIの出力の不安定さは
  - モデル性能だけの問題ではない
- **AIに与える前提知識の設計**が重要

具体的なアドバイス

- APIに投げるプロンプト内で
  - *Lean 4 の定義・記法の前提を明示*
- 技術解説ページ（指定URL）を
  - 参照させることで解釈のぶれを抑える

③学びと今後の展望
― 視点の変化とこれからの活用

Cコースだから得られた学び

テーマもゴールも自分で決める

学びは一律ではなく、挑戦次第

正解が用意されていない環境

未知の可能性が最も大きいコース

事前 → 事後で変わったAIへの向き合い方

事前：
課題に対してどのAIモデル・機能・ツールを使うかを重視

・AIのタイプや性能の比較が中心
・Cコースでの試行錯誤と助言

事後：
プロンプトや前提知識を含めた設計を重視

・利用者視点から提供者視点へ

今後への活用と展望

利用者 × 提供者の両視点

安定性・公平性を意識した設計

教育・研究・開発への応用

オープンに学び続ける姿勢

ご清聴ありがとうございました